Linking Automated Data Analysis and Visualization with Applications in Developmental Biology and High-energy Physics

Vom Fachbereich Informatik der Technischen Universität Kaiserslautern zur Erlangung des akademischen Grades Doktor der Naturwissenschaften (Dr. rer. nat.) genehmigte Dissertation von

Oliver Rübel.

Abstract

Knowledge discovery from large and complex collections of today's scientific datasets is a challenging task. With the ability to measure and simulate more processes at increasingly finer spatial and temporal scales, the increasing number of data dimensions and data objects is presenting tremendous challenges for data analysis and effective data exploration methods and tools. Researchers are overwhelmed with data and standard tools are often insufficient to enable effective data analysis and knowledge discovery.

The main objective of this thesis is to provide important new capabilities to accelerate scientific knowledge discovery form large, complex, and multivariate scientific data. The research covered in this thesis addresses these scientific challenges using a combination of scientific visualization, information visualization, automated data analysis, and other enabling technologies, such as efficient data management. The effectiveness of the proposed analysis methods is demonstrated via applications in two distinct scientific research fields, namely developmental biology and high-energy physics.

Advances in microscopy, image analysis, and embryo registration enable for the first time measurement of gene expression at cellular resolution for entire organisms. Analysis of high-dimensional spatial gene expression datasets is a challenging task. By integrating data clustering and visualization, analysis of complex, time-varying, spatial gene expression patterns and their formation becomes possible. The analysis framework MATLAB and the visualization have been integrated, making advanced analysis tools accessible to biologist and enabling bioinformatic researchers to directly integrate their analysis with the visualization.

Laser wakefield particle accelerators (LWFAs) promise to be a new compact source of high-energy particles and radiation, with wide applications ranging from medicine to physics. To gain insight into the complex physical processes of particle acceleration, physicists model LWFAs computationally. The datasets produced by LWFA simulations are (i) extremely large, (ii) of varying spatial and temporal resolution, (iii) heterogeneous, and (iv) high-dimensional, making analysis and knowledge discovery from complex LWFA simulation data a challenging task. To address these challenges this thesis describes the integration of the visualization system VisIt and the state-of-the-art index/query system FastBit, enabling interactive visual exploration of extremely large three-dimensional particle datasets. Researchers are especially interested in beams of high-energy particles formed during the course of a simulation. This thesis describes novel methods for automatic detection and analysis of particle beams enabling a more accurate and efficient data analysis process. By integrating these automated analysis methods with visualization, this research enables more accurate, efficient, and effective analysis of LWFA simulation data than previously possible.

Zusammenfassung

Die Analyse komplexer wissenschaftlicher Daten ist eine zentrale Herausforderung in der heutigen Forschung. Komplizierte Vorgänge können immer detaillierter gemessen und simuliert werden. Dies führt dazu, dass aktuelle wissenschaftliche Datensätze Informationen über immer mehr Datendimensionen und Datenobjekte enthalten. Wissenschaftler werden überwältigt mit Daten, und herkömmliche Analyseverfahren sind oft nicht ausreichend um eine effektive Analyse der vorhandenen Daten zu ermöglichen.

Das Hauptziel dieser Dissertation ist es, wichtige neue Methoden zu entwickeln um den effizienten Gewinn von Wissen von großen, komplexen, multivariaten wissenschaftlichen Daten zu ermöglichen. Um dieses Ziel zu erreichen, verwendet diese Arbeit eine Kombination aus Visualisierung, Datenanalyse, und anderen grundlegenden Technologien (zum Beispiel Verfahren zum Management von großen Daten). Die Effektivität der vorgeschlagenen Methoden wird demonstriert anhand zweier unterschiedlicher Applikationen aus der Entwicklungsbiologie und der Hochenergiephysik.

Fortschritte in der Mikroskopie, Bildanalyse, und Verfahren zur Registrierung von Embryonen ermöglichen zum ersten Mal die Messung von Genexpression mit zellulärer Auflösung für vollständige Organismen. Räumliche Genexpressionsdaten sind hochdimensional, wobei die verschiedenen Gene durch komplexe Prozesse miteinander in Verbindung stehen. Die Analyse dieser hochkomplexen Daten ist schwierig. Durch die Integration von Clustering und modernen Visualisierungsverfahren ermöglicht diese Forschungsarbeit die Analyse von komplexen, räumlichen und sich zeitlich verändernden Genexpressionsmustern sowie deren Beziehungen zueinander. Mittels der Integration von MATLAB, einem fortgeschrittenen System zur Datenanalyse, mit der Visualisierung ermöglicht diese Forschungsarbeit Biologen den Zugriff auf fortgeschrittenen Analyseverfahren.

Plasmabasierte Kielfeld-Beschleuniger ermöglichen die Erzeugung von hochenergetischen Partikeln in sehr kurzer Distanz und versprechen in Zukunft die Möglichkeit des Baus von wesentlich kompakteren Partikelbeschleunigern, die Anwendung in der Medizin wie auch in der Physik finden können. Rechnergestützte Simulationen werden verwendet, um ein besseres Verständniss der komplizierten physikalischen Vorgänge zu erlangen. Die resultierenden Simulationsdaten sind (i) extrem groß, (ii) haben unterschiedliche zeitliche und räumliche Auflösung, (iii) sind heterogen und (iv) hochdimensional. Die Analyse solcher hochkomplexen Daten ist kompliziert. Durch die Integration des Visualisierungssystems VisIt mit dem Datenmanagement-system FastBit ermöglicht diese Forschungsarbeit die interaktive visuelle Analyse von extrem großen dreidimensionalen Simulationsdaten. Im Rahmen der Datenanalyse sind hier insbesondere charakteristische, hochenergetische Partikelbündel, die während der Simulation geformt werden, von Interesse. Um einen effizienten Datenanalyseprozess zu ermöglichen, wurden im Rahmen dieser Dissertation verschiedene Verfahren zur automatischen Detektion und Analyse von Partikelstrahlen entwickelt. Die Integration dieser automatischen Verfahren mit der Visualisierung ermöglicht die genaue, effiziente, und effektive Analyse von Simulationsdaten von Kielfeld-Beschleunigern.

Acknowledgments

I want to use this opportunity to thank all people who made this work possible. First, I would like to thank my advisors Bernd Hamann and Hans Hagen. Bernd and Hans gave me the opportunity to conduct my research at world-renowned research groups: (i) the International Research Training Group "Visualization of Large and Unstructured Data Sets – Applications in Geospatial Planning, Modeling, and Engineering" (IRTG 1131), University of Kaiserslautern, Germany; (ii) the Institute for Data Analysis and Visualization (IDAV), University of California, Davis, CA, USA; and (iii) the Visualization Group, Lawrence Berkeley National Laboratory (LBNL), Berkeley, CA, USA. I would also like to thank E. Wes Bethel for his financial support, advice, and guidance. Furthermore, I would like to thank Gunther H. Weber for his support, advice, and help throughout my work.

During my time as a doctoral student I had the chance and honor to collaborate with many outstanding scientist; my special thanks go to all the great people who supported this work. My thanks go to the members of the Visualization Group at LBNL: E. Wes Bethel, Hank Childs, Janet Jacobsen, Prabhat, Daniela Ushizima, and Gunther H. Weber. My thanks go to the members of the Berkeley Drosophila Transcription Network Project (BDTNP) at LBNL: Mark Biggin, Angela DePace, Michael B. Eisen, Charless C. Fowlkes, Clara Henriquez, Min-Yu Huang, Soile S.V.E. Keränen, David Knowles, Cris Luengo Hendriks, Jitendra Malik, and Lisa Simirenko. My thanks go to the members of the Lasers, Optical Accelerator Systems Integrated Studies (LOASIS) Program at LBNL; in particular, I would like to thank Cameron Guy Robinson Geddes and Estelle Cormier-Michel. My thanks go to the members of the SciDAC Scientific Data Management Research Group (SDM) at LBNL for their support of FastBit; in particular, I would like to thank Kesheng "John" Wu. My thanks go to Jeremy Meredith and Sean Ahern from the Oakridge National Laboratory (ORNL). My thanks go to Peter Messmer from Tech-X. My thanks go to the members of the IRTG 1131 at the University of Kaiserslautern, Germany, for their help and support; in particular, I would like to thank Inga Scheler, Peter Dannenmann, and Mady Gruys. My thanks go to the members of the Institute for Data Analysis and Visualization (IDAV) at the University of California, Davis, for their support. My thanks go to the VisIt development team for their continued efforts to develop, maintain, and improve VisIt. My thanks go to the VORPAL development team for ongoing efforts to develop and maintain this parallel C++ framework on a variety of supercomputing platforms including those at the National Energy Research Scientific Computing Center(NERSC).

This work was supported by the International Research Training Group "Visualization of Large and Unstructured Data Sets – Applications in Geospatial Planning, Modeling, and Engineering," at the University of Kaiserslautern, Germany.

The work on analysis of 3D gene expression data was supported by the National Institutes of Health through grant GM70444, by the National Science Foundation through award ACI 9624034 (CAREER Award), through the Large Scientific and Software Data Set Visualization (LSSDSV) program under contract ACI 9982251, by the Lawrence Berkeley National Laboratory (LBNL) Laboratory Directed Research Development (LDRD) program; and by an Information Technology Research (ITR) grant. Work at LBNL is conducted under Department of Energy contract DE-AC02-05CH11231.

The work on analysis of laser wakefield accelerator simulation data was supported by the Director, Office of Science, Offices of High Energy Physics and Advanced Scientific Computing Research, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 through the Scientific Discovery through Advanced Computing (SciDAC) program's Visualization and Analytics Center for Enabling Technologies (VACET) and the COMPASS project. This research used resources of NERSC, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

Finally, and most importantly, I would like to thank my parents, Hans-Werner and Renate, and my sister Anja, for their love, constant support, and patience during the long years of my eduction. Although great distances have kept us apart, without their ongoing encouragements and confidence, this dissertation would have not been possible. I would also like to thank Rachel for her love, ongoing encouragement and confidence, and for keeping me in good spirit. My thanks also go to my grandmother Ida, my nephew Lars, and my whole extended family for all the joy and for being part of my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Knowledge discovery from large and complex collections of today's scientific datasets is a challenging task. Due to advances in data acquisition and scientific computing, today's datasets become increasingly complex. With the ability to measure and simulate more processes at finer scales the number of data dimensions and data objects has increased significantly in today's scientific datasets while the phenomena researchers are able to investigate become increasingly complex. Researchers are overwhelmed with data and standard tools are often insufficient to enable efficient data analysis and, hence, discovery of information and knowledge from the data.

This thesis provides important new capabilities to accelerate scientific knowledge discovery form large, complex, multivariate scientific data. The research presented addresses these scientific challenges using a combination of scientific visualization, information visualization, automated data analysis, and other enabling technologies such as efficient data management.

Advances in microscopy, image analysis, and embryo registration enable for the first time measurement of gene expression at cellular resolution for entire organisms. Using the fruitfly *Drosophila melanogaster* as model organism, the Berkeley Drosophila Transcription Network Project (BDTNP) has developed a novel data acquisition pipeline enabling researchers to analyze the activity of up to several thousand genes at cellular resolution at currently up to six developmental stages (i.e., timesteps).

Analysis of this novel type of data is a challenging task, in particular due to the large amount of genes (i.e., data dimensions) and the complex interactions between the many genes. While visualization is a powerful approach to gain deeper insights into such complex datasets, it is limited in this case because the intricate and often subtle nature of 3D gene expression data makes visual detection of all existing features very difficult. Data clustering has already proven to be very powerful at revealing details from conceptually simpler forms of expression data, such as that from microarray experiments, that are not easily detected visually in raw data. Appropriately defining clustering parameters as well as validation and interpretation of clustering results, however, is a non-trivial endeavor.

By integrating automated data analysis using data clustering and state-of-the-art visualization using the visualization system PointCloudXplore (PCX), this research effort has developed new approaches that overcome the difficulties in both visual analysis and data clustering and enable more detailed analysis of 3D gene expression data than previously possible. By providing an interface between PCX and MATLAB, an advanced data analysis environment, this research provides researchers direct access to advanced bioinformatics analysis tools enabling more efficient and accurate analysis. To demonstrate the effectiveness of the developed system, the proposed framework is used to: (i) define spatial boundaries of gene expression patterns, (ii) characterize the temporal variation of genes, and (iii) analyze how mRNA patterns are controlled by their regulatory transcription factors.

The main challenge in the context of 3D gene expression data is given by the large number of data dimensions and the complex relationships between them; however, the challenges in other scientific research areas may be quite different. Due to the availability of ever more powerful supercomputers and advances in modern scientific computing, simulation of increasingly complex phenomena at very fine scales becomes possible. Besides complexity of the phenomena under investigation, the main challenges in the context of high-end numerical simulations often stem from the amount of data produced by the simulations. Efficient methods for data display as well as novel data analysis methods are required to enable researchers to effectively investigate and derive new knowledge from large collections of data. Whereas the data itself is extremely large, the phenomena and features of interest are often very small making data analysis an ever more challenging task. Manual detection of the features of interest is time-consuming and in many cases impossible. Automated methods supporting feature detection in extremely large datasets promise to enable a more focused and efficient analysis process enabling analysis of even extremely large simulated datasets. To demonstrate the effectiveness of the proposed integrated analysis approach based on linking of data analysis and visualization also in the context of large simulation data, this research effort applies the described concepts to the analysis of laser wakefield particle accelerator (LWFA) simulation data.

LWFAs promise to be a new compact source of high-energy particles and radiation, with wide applications ranging from medicine to physics. Laboratory experiments are expensive, require careful planning, and in practice it is usually impossible to record the complete evolution of LWFA experiments. Physicists, therefore, model LWFAs computationally to gain insight into the complex physical processes.

The datasets produced by LWFA simulations are (i) extremely large, (ii) of varying spatial and temporal resolution, (iii) heterogeneous, and (iv) high-dimensional, making analysis and knowledge discovery from complex LWFA simulation data a challenging task. One main feature researchers are interested in are beams of high-energy particles formed during the course of LWFA simulations. To enable efficient and accurate analysis of these particle beams, dedicated mechanisms for selection and detection of particle beams are needed. Furthermore, to support analysis of selected particle bunches of interest, effective visualization methods are required.

To enable efficient visual analysis and interactive selection of particle beams in extremely large and complex LWFA simulation datasets, this research describes the integration of the advanced visualization system VisIt and state-of-the-art data management using FastBit. To provide the user with an effective interface for performing data selection operations (e.g., to define particle beams), this thesis describes several new extensions to histogram-based parallel coordinates. The combination of parallel coordinates as an interface for data selection and advanced scientific visualization methods enables more accurate and efficient analysis of LWFA simulation data than previously possible. This approach, while applied here to accelerator science, is generally applicable to a broad set of scientific applications, and is implemented in a production-quality visual data analysis infrastructure. While visual analysis of particle beams based on interactive selection is effective, it requires substantial manual input from the user and may, hence, be time-consuming. Automating the most time-consuming parts of the data analysis process promises to support a more focused and efficient analysis process enabling the analysis of large collections of LWFA simulation datasets. This thesis presents (i) *automatic beam detection*, a method aimed at identifying the highest energy particle bunch in an LWFA simulation dataset, and, (ii) *automatic beam path analysis*, a method aimed at identification and comparison of multiple particle bunches. By integrating these automatic analysis methods with the visualization, efficient analysis of LWFA simulation datasets becomes possible. The proposed framework is applied to address relevant issues, such as (i) detection and definition of particle beams, (ii) analysis of the formation and acceleration of particle beams, (iii) investigation of the quality of particle beams, and (iv) comparison of the behavior and quality of multiple particle beams, demonstrating the effectiveness of the developed system.

1.1 Framework for Knowledge Discovery from Scientific Data

While the challenges in the selected application areas are quite different, the basic analysis frameworks used for knowledge discovery from these complex types of data are similar. Figure 1.1

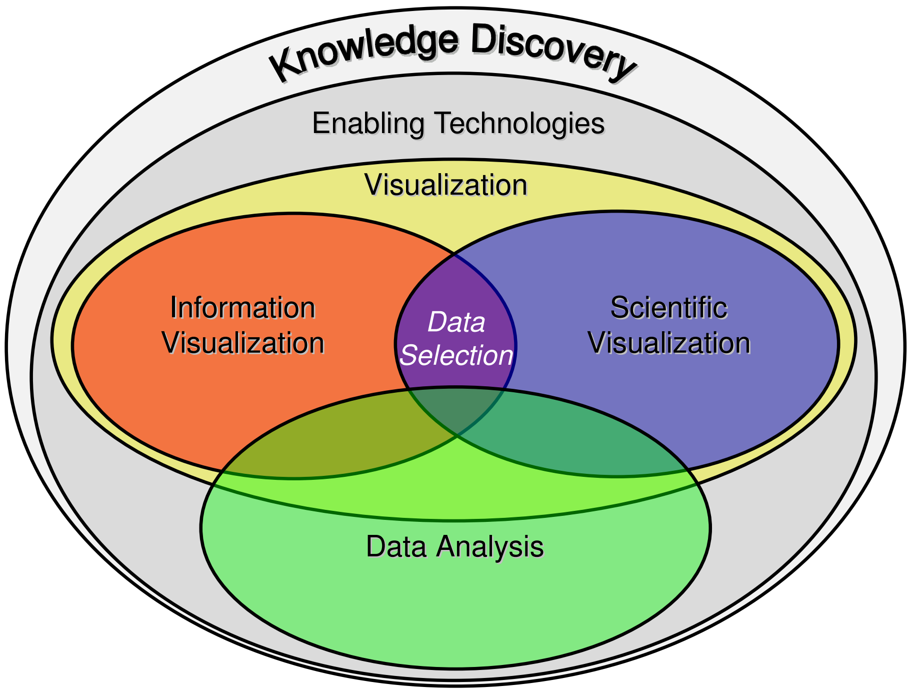


Figure 1.1: Overview of the basic components of the developed analysis framework for knowledge discovery from scientific data.

illustrates the basic compontents of the proposed analysis framework consisting of (i) enabling technologies, (ii) visualization, and (iii) data analysis.

Enabling technologies are fundamental methods required to enable data analysis and knowledge discovery that may not necessarily be part of the analysis itself. For example, in the case of extremely large datasets efficient methods for data access are essential to enable effective data analysis. In the context of large data collections, efficient data management methods, such as online databases, are required.

To enable effective data exploration, this research makes use of a set of specific scientific and information visualization methods. Information visualization views are used for visual exploration of variable space and to identify relationships between different data dimensions. Views of high-dimensional variable space also serve as vehicles for defining advanced queries, enabling accurate selection of complex data features. Advanced scientific visualization views support detailed analysis of physical data characteristics. Different views —each highlighting different aspects of the data— are linked via the concept of data selection (brushing). Selected data subsets can be highlighted in any view enabling detailed analysis and knowledge discovery from complex scientific data.

While interactive data exploration based on linked multiple views is effective, it also has limitations. Visual detection of all features, and in particular fine and subtle features, is often impossible. Furthermore, manual exploration may be a time-consuming process hindering the analysis of large data collections. Automated data analysis methods promise to overcome these limitations in visual data analysis by automating the most complex and time-consuming steps of the analysis pipeline. For example, data clustering allows identification of cells in an embryo that show similar behavior in gene expression. In the context of LWFA simulation data, efficient data analysis methods enable automated detection of particle beams and analysis of their temporal evolution. In practice, interpretation of automated analysis results can produce wrong interpretation or might not be intuitive to comprehend, and proper definition of analysis parameters is often complicated.

By linking automated data analysis and visualization, this research overcomes the difficulties in both visualization and automated data analysis. Automating the detection of relevant data features supports a more detailed and focused data analysis and enables development of visualizations that focus on the main data portions of interest, significantly reducing visual clutter and occlusion of important information. By defining input parameters of data analysis procedures based on visualization, more accurate and intuitive use of automated data analysis methods becomes possible. Visualization also provides means for validation, interpretation, and investigation of results generated by automated data analysis algorithms. Visualization combined with automated data analysis enables accurate and efficient analysis of large and complex scientific data and effectively supports knowledge discovery.

1.2 Overview and Contributions

The following parts of this manuscript are structured as follows. Chapter 2 reviews the state of the art in data analysis and visualization relevant to this work. Chapter 3 provides and introduction to 3D gene expression and laser wakefield particle acceleration and describes the relevant background in developmental biology and physics, respectively. Starting with Chapter 4, only new results are presented.

Chapter 4 describes the developed framework for knowledge discovery from 3D gene expression data. Sections 4.1 and 4.2 provide an overview of the main problem and the state of the art in visual analysis of 3D gene expression data. Section 4.3 provides an overview of the proposed analysis framework which is explained in detail afterwards in Sections 4.4 to 4.7. The specific scientific contributions made by this dissertation are the:

(i) application of data clustering to 3D gene expression data (Section 4.4.1),

(ii) evaluation of the number of clusters k in the context of 3D gene expression clustering (Section 4.4.2),

(iii) integration of data clustering and visualization into one framework and improvement of overall analysis quality via dedicated post-processing of clustering results based on visualization (Section 4.5),

(iv) integration of the visualization system PointCloudXplore (PCX) with MATLAB (Section 4.6), and

(v) concerning specific application areas, Section 4.7 discusses the use of the proposed framework to: (a) objectively define spatial boundaries of gene expression patterns, (b) characterize the temporal variation of genes, and (c) to analyze how mRNA patterns are controlled by their regulatory transcription factors.

Chapter 5, describes the developed framework for knowledge discovery from LWFA simulation data. Sections 5.1 and 5.2 provide an overview of the main problem and the state of the art in analysis of LWFA data. Section 5.3 provides an overview of the proposed analysis framework which is explained in detail afterwards in Sections 5.4 to 5.8. As specific high-level contributions of this thesis, Chapter 5 describes:

(i) a novel framework for high-performance visual data exploration of extremely large data based on the integration of the visualization system VisIt and the index/query system FastBit (Section 5.4),

(ii) automatic beam detection; a novel approach for automatic detection of the highest-energy particle bunch based on bunch lifetime analysis and fuzzy clustering (Section 5.5),

(iii) automatic beam path analysis; a novel approach for automatic detection and analysis of multiple particle beams based on the complete temporal path of the particles that form them (Section 5.6),

(iv) integration of the automatic analysis with visualization to enable efficient analysis of extremely large 3D particle datasets (Section 5.7), and

(v) concerning specific application areas, Section 5.8 discusses the use of the proposed framework to address relevant issues, such as (a) detection and definition of particle beams, (b) analysis of the formation and acceleration of particle beams, (c) investigation of the quality of particle beams, and (d) comparison of the behavior and quality of multiple particle beams.

Chapter 6 concludes this thesis with a summary of the described research and also points out possible directions for future research.

Chapter 2

State of the Art: Data Analysis and Visualization

This chapter introduces relevant background material and related work in data analysis and visualizations. This research makes use of methods from scientific visualization, information visualization, as well as data analysis. It is beyond the scope of this thesis to provide a complete overview of these large research fields. The goal of this chapter is rather to provide an introduction to the various related research areas and the main methods employed in the context of this thesis. The different main parts of this chapter first provide a high-level overview of the scope of the different related research fields and afterwards focus on the specific methods relevant for this work.

Sections 2.1 and 2.2 first provide an introduction to scientific visualization and information visualization and in particular describe the main visualization techniques used in the context of this work. Section 2.3 describes the linking of scientific and information visualization views via the concept of brushing (i.e., data selection). Having introduced the main visualization concepts, Section 2.4 describes the visualization system VisIt. VisIt is used in this work for visualization and analysis of laser wakefield particle accelerator (LWFA) simulation data. Afterwards, the concept of query-driven visualization (QDV), used here to enable efficient visual exploration of LWFA data, is described in Section 2.5. QDV relies on efficient data management to enable fast data exploration via a series of data queries. Section 2.6 introduces FastBit, an index/query system used in the context of this thesis for implementation of the QDV concept in VisIt. This chapter concludes with an introduction to data classification, i.e., methods used in the context of this thesis for automatic identification of relevant data features (see Section 2.7).

2.1 Scientific Visualization

Over the last three decades, many different definitions of scientific visualization have been introduced in the literature. R.A. Earnshaw [1], e.g., defined scientific visualization as follows: “Scientific visualization is concerned with exploring data and information in such a way as to gain understanding and insight into the data. The goal of scientific visualization is to promote a deeper level of understanding of the data under investigation and to foster new insight into the underlying processes, relying on the humans’ powerful ability to visualize. In a number of instances, the tools and techniques of visualization have been used to analyze and display large volumes of, often time-varying, multidimensional data in such a way as to allow the user to extract significant features and results quickly and easily.”(Chapter 1 in [1]). An alternative definition by B. McCormick, T. DeFanti, and M. Brown [2] describes visualization as follows: “Visualization is a method of computing. It transforms the symbolic into the geometric, enabling researchers to observe their simulations and computations. Visualization offers a method for seeing the unseen. It enriches the process of scientific discovery and fosters profound and unexpected insights. In many fields it is already revolutionizing the way scientists do science.”(in [2]). A list of various other definitions of visualization can be found online [3].

The main objective of visualization is to enable data understanding and knowledge discovery. The discipline of scientific visualization focuses on the study of the representation of large collections of scientific numerical data in a way that it can be perceived by humans. Scientific data commonly has a physical meaning and an inherent spatial structure. Most scientific visualization methods, therefore, focus on visualization in physical space. Scientific visualization predominantly concentrates on the development and use of graphical techniques to facilitate data analysis and understanding but also other perceptual stimuli, such as sound and haptic force feedback approaches, have been incorporated successfully in the visualization process [4, 5].

A typical visualization algorithm can be thought of as a transformation from one data form into another, e.g., from a scalar value to color (see Chapter 1, [6]). A visualization algorithm may transform (i) the geometry of the data (e.g., via scaling, translation, or rotation of the data) , (ii) the topology of the data (e.g., by converting the data from a polygonal to an unstructured grid), (iii) the attributes of the data (e.g., by computing derived quantities such as vector magnitude), or (iv) both dataset structure and attribute data (e.g., by computing contour lines or surfaces). Visualization algorithms are often classified based on the type of data attributes they operate on, such as scalar, vector, or tensor algorithms.

Scalar data fields define a single numerical value associated with each point and/or cell of a dataset (e.g., temperature or velocity magnitude). The most common algorithm for visualization of scalar data include color mapping, contouring, and direct volume rendering (see Section 2.1.1 for details).

Flow field vector data, for example, describes a two-dimensional or three-dimensional represention of direction and maginitude, e.g., an air-flow around an airplane. Vector fields are often visualized via oriented glyphs (e.g., hedgehogs), warping or displacement of geometry, animation (e.g., by showing the movement of particles in a flow), as well as streamlines and pathlines.

Tensors are commonly represented as a matrix in the context of scientific visualization and can be thought of as a generalization of the vector concept. Tensors are used, e.g., to describe the stress in a three-dimensional material — essentially a set of forces of varying strength acting in all directions within a material — and are often visualized via tensor glyphs.

Besides scalar, vector, and tensor visualization algorithms, there are other techniques that are applicable to a broader range of data types (such as cutting, slicing, selection, probing, or subsampling) as well as methods that utilize a combination of different types of data (e.g., glyphs that are deformed based on stress and colored by temperature). In the taxonomy of visualization algorithms described by Schroeder and Martin, such visualization algorithms are referred to as modeling algorithms (Chapter 1, [6]). An overview of the most common visualization algorithms is provided in [6].

For this work, algorithms for visualization of scalar fields are the most relevant. Section 2.1.1 introduces the most common scientific scalar field visualizations used in the context of this work.

2.1.1 Scalar Field Visualization

In visualization, color is commonly used for display of membership (i.e., to which group or cluster does a datum belong to) or to encode scalar values. *Color mapping* is one of the most common scalar field visualizations. Color mapping algorithms map scalar data to colors displayed via the standard coloring and shading capabilities of graphics libraries and hardware. Commonly, color mapping is implemented via a color lookup table where the scalar values serve as indices into the color table. The key to a good color mapping is the proper choice of a color lookup table [7]. Information may be encoded via various aspects of color, such as color value (e.g., red, green, blue), brightness or saturation of color, as well as transparency. Colors are often mapped onto two- or three-dimensional objects (such as a wing or car in an airflow) to enhance the visualization. Figure 2.1a shows an example visualization of a scalar field with colors mapped onto the outer faces of the cube-like volume the scalar field is defined in. The color map used here is similar to the common rainbow color map, where blue=low, green=medium, and red=high. In addition to the value of color, the brightness of color is varied periodically to achieve higher

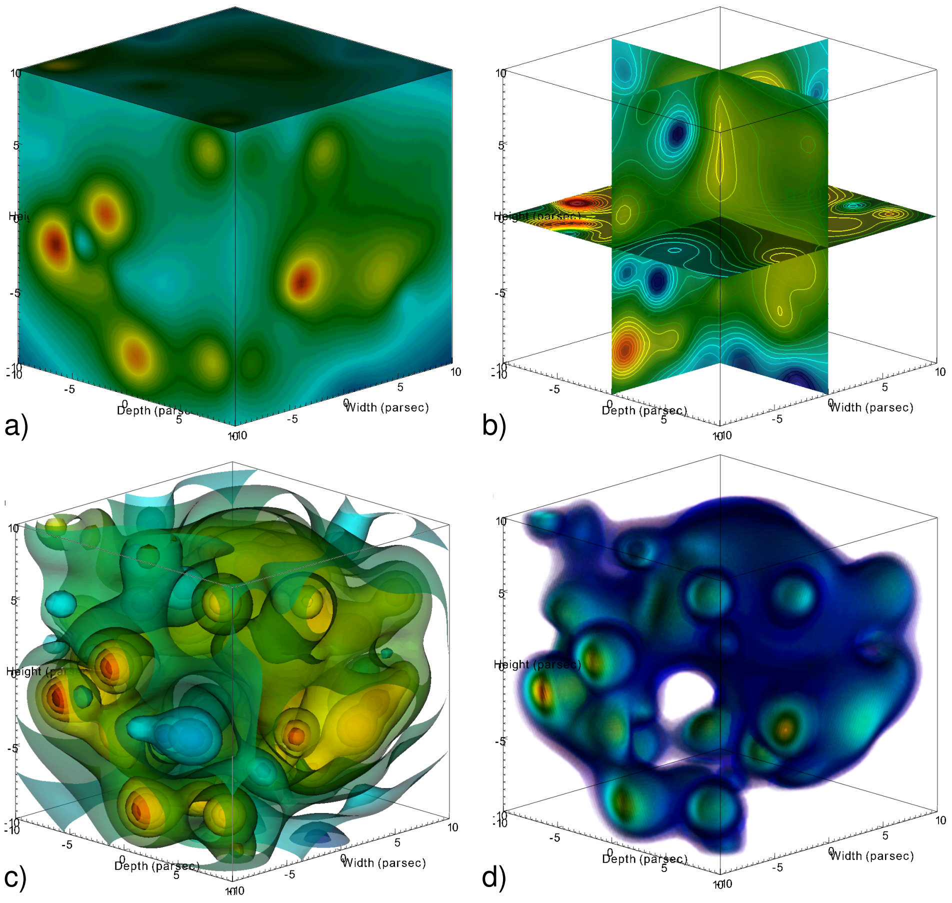


Figure 2.1: Visualizations of a three-dimensional (3D) scalar field using: a) Color mapping, b) Three slicing planes colored according to the scalar values and additional contours, c) Semi-transparent isosurfaces, and d) direct volume rendering. In the color mapping used in the plots red=high, green=medium, and blue=low scalar values. The visualizations were created using the visualization system VisIt (see Section 2.4) and show the *hardyglobal* scalar field of the VisIt example dataset *noise.silo*.

contrast.

To be able to investigate inner structures of a volumetric scalar field, one often cuts through the volume using a surface. The interpolated data values are displayed on the cutting surface. This technique is in general referred to as *cutting*. In case that a plane is used as cutting surface the alternative term *slicing* is often used instead. In case of vector data the cutting surface may also be warped or displaced based on interpolated vector values. Figure 2.1b shows an example where the dataset shown in Figure 2.1a is sliced using three axes-aligned planes centered in the volume. Scalar values are displayed on the planes using the same color mapping as in Figure 2.1a. Contours are shown in addition on the slicing planes to highlight the structure of the

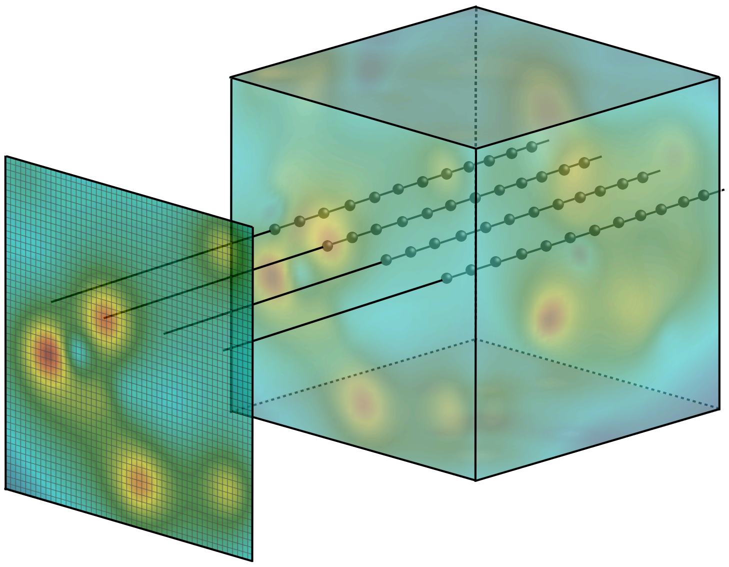


Figure 2.2: Illustration of direct volume rendering (DVR). From the viewer's eye point, and passing through a pixel in the image plane a set of rays is cast through the volume. Along each ray sample points (black spheres) are interpolated to determine the accumulated color value of the corresponding pixel.

scalar field.

Contouring algorithms construct the boundary surface (or line) between different regions of a scalar field by computing a surface of constant scalar value (i.e., an isosurface). In the context of weather forecast maps contours are commonly used to show the separation between regions of high and low pressure via so-called *isobars* or the separation between regions of varying temperature via so-called *isotherms*. A single boundary can be expressed via a *n*-dimensional iso-value function $F(x_1, x_2, ..., x_n) = c$ where c is the contour value. Such a contour reveals the separation between regions with $F(x_1, x_2, ..., x_n) < c$ and $F(x_1, x_2, ..., x_n) > c$. When using color mapping for display of scalar fields, a human's eye often separates colored areas into distinct regions (see Figure 2.1b). Contouring effectively constructs these boundaries and, hence, supports analysis of the structure of scalar fields. Figure 2.1c shows an example isosurface visualization of a three-dimensional scalar field. An overview of a variety of algorithms for isosurface computation is provided in [6].

While contour-based visualizations are often useful, they may require large amounts of geometry (e.g., triangles) to accurately represent a surface contained in a volume. Also, contours are not well-suited for visualization of amorphous phenomena, such as fog, fire, or clouds. *Volume rendering* is a technique that allows creation of a two-dimensional image directly from three-dimensional volumetric data. The basic principle of volume rendering is illustrated in Figure 2.2. For each pixel of the image plane a ray is cast through the volume. Along each ray a set of sample points is obtained via interpolation. The information of the sample points along a ray is combined to determine the accumulated color value of the corresponding pixel. Depending on how the samples along a ray are combined, different volume rendering modes are created, e.g., (i) x-ray rendering mode (sum of samples along the ray), (ii) maximum intensity projection (maximum sample value found along a ray), and (iii) full volume rendering (the interpolated samples are processed to simulate the light transport within a volumetric medium) [6]. Volume rendering can be achieved using various techniques, such as image-order methods (via ray-casting as described here), object-order methods (e.g., splatting), or domain-based methods. An overview of volume rendering is provided in [6] (Part III). Figure 2.1d shows an example volume rendering of the same dataset used for illustration of color mapping, cutting, and contouring.

2.2 Information Visualization

Information visualization focuses on the study of the visual representation of large collections of non-numerical information. Information visualization is used typically for the analysis of large collections of text (e.g., source code), social networks, or business data. Similar to scientific visualization, information visualization focuses on the development and use of graphical techniques to facilitate data analysis and understanding. In contrast to scientific visualization, however, information visualization focuses on abstract data with no physical meaning or inherent spatial structure, such as unstructured text or high-dimensional point data. Today's scientific data is becoming increasingly complex, large, and high-dimensional. Information visualization methods have, therefore, found in recent years increasing interest also in the context of scientific visualization. In the context of scientific visualization, abstract information visualization views are commonly used in combination with physical views of the data (see Section 2.3).

The following parts of this section introduce various information visualization views used in the context of this work. Scatter plots (see Section 2.2.1) and parallel coordinates (see Section 2.2.2) are two common approaches for the display of abstract, high-dimensional data. Descriptive statistic plots are used to summarize statistical properties of the data and to investigate data distributions (see Section 2.2.3).

2.2.1 Scatter Plots

Scatter plots are a common visualization technique in statistics, information visualization, as well as scientific visualization [8]. Scatter plots are conceptually the simplest way to visualize

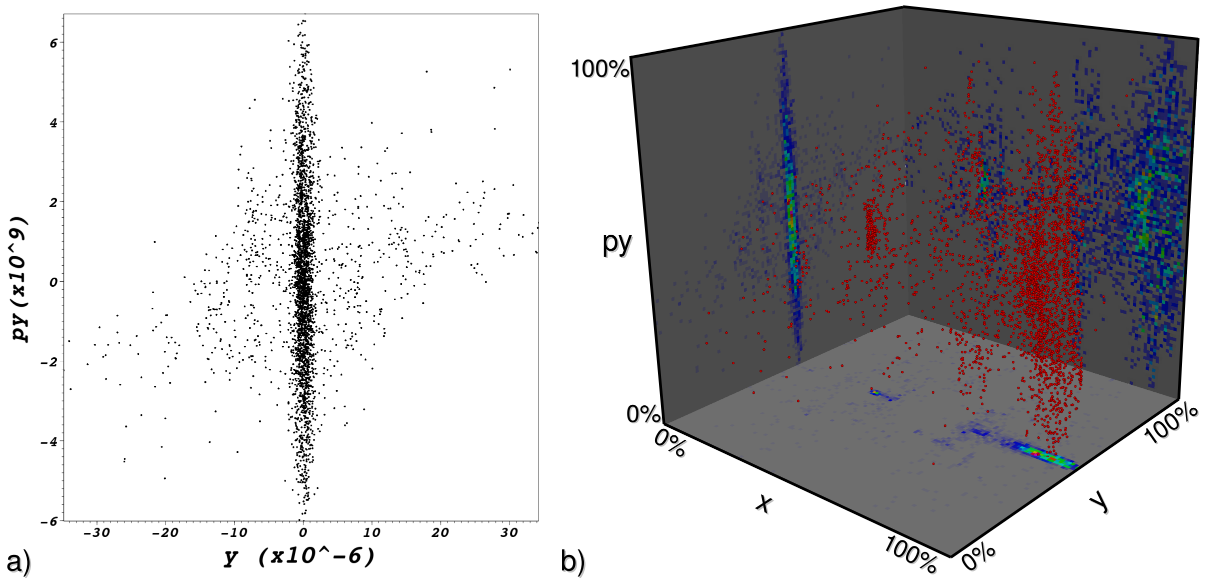


Figure 2.3: a) Illustration of a two-dimensional (2D) scatter plot in (y, py) space of all accelerated particles (i.e., particles with $px > 10^{10}$) at a single time step of a laser wakefield particle accelerator (LWFA) simulation in 2D (see Section 3.2 for an introduction to LWFAs). b) Illustration of a three-dimensional (3D) scatter plot in (x, y, py) space of the same data as shown in *a*. The data points of the scatter plot (each representing a particle) are shown via red spheres. The corresponding 2D histograms are shown in color on the outside surfaces of the cube defined by the scatter plot (gray/blue=low , green=medium, and red=high particle density).

relationships between two data variables. Two selected variables are mapped to the axes of a Cartesian coordinate system with each axis ranging from the corresponding variable's minimum (at the origin) to the maximum value (see Figure 2.3). Each discrete datum is represented by a single point in the scatter plot. The location of a data point is specified based on the values of the corresponding data record in the data dimensions displayed. This concept can be directly extended to three-dimensional (3D) space (see Figure 2.3). For higher-dimensional data a series of two-dimensional scatter plots is usually arranged in the form of a matrix showing the scatter plots of all possible variable pairs (see Figure 2.4 left).

Color and shape of data points are often used to improve the visualization or to display additional information. Transparency is used to improve depth perception as well as to ease detection of density structures in scatter plots of large data sets. Piringer et al. [9] introduced a variety of enhancements to 3D scatter plots, improving depth perception and perception of the sample distribution in all dimensions. By varying color as well as size of data points depending on the distance to the viewer, depth perception is improved. Halos (a thin circle around each point) allow individual data points to be distinguished more easily. Scatter plots are often combined with histogram plots to ease detection of density patterns. In the 2D case, the corresponding 1D histograms may be shown on the side of each axis. Similarly, 2D histograms can be mapped to the outer faces of the cube defined by a 3D scatter plot [9]. Due to possible occlusion of the actual scatter plot, the display of additional histograms may, however, be less effective in the 3D case.

Recently Bachthaler and Weiskopf introduced continuous scatter plots as a generalization of the scatter-plot concept to the visualization of spatially continuous data [10]. Rather than by a collection of discrete data points, the data is represented by a continuous dense plot that considers the interpolation defined between the discrete data samples.

2.2.2 Parallel Coordinates

Parallel coordinates are a common information visualization technique for high-dimensional data sets. In the context of this work, parallel coordinates are used for data display as well as for performing data-mining operations based on multi-dimensional thresholding. Parallel coordinates are believed to be originally invented by Maurice d'Ocagne in 1885 [11] and were later independently re-discovered and popularized by Inselberg [12] and Wegman [13]. This section provides a brief introduction to parallel coordinates and several extensions to standard parallel coordinates. For a detailed introduction to modern parallel coordinates and their applications see [14, 15].

In parallel coordinates, each data variable of a multivariate dataset is represented by one axis. The parallel coordinates plot is constructed by drawing for each data record a polyline connecting the points where the data record's variable values intersect each axis. As illustrated in Figure 2.5(top), there is a duality between the Cartesian coordinates as used in a scatter plot and the parallel coordinate system. A two-dimensional point in Cartesian coordinates maps to a line segment in parallel coordinates and vice versa. A n-dimensional point in Cartesian space corresponds to a poly-line consisting of $n-1$ line segments in parallel coordinates.

One main advantage of parallel coordinates is that they enable visualization of multi-dimensional data in a single 2D view, rather than multiple independent views such as in a scatter plot matrix (see Figure 2.4). Within the limitations of the display used, parallel coordinates scale well with increasing number of data dimensions, i.e., one simply needs to add an additional parallel axis to add a new data dimension. The number of data dimensions that can effectively be displayed is mainly limited by the available screen space and the required minimum distance between neighboring axes of the plot. Parallel coordinates have also proven to be an effective tool for data mining. In particular, parallel coordinates can be used to define multi-dimensional range queries simply by adding two sliders to each parallel axis. Using these sliders a user can define the data

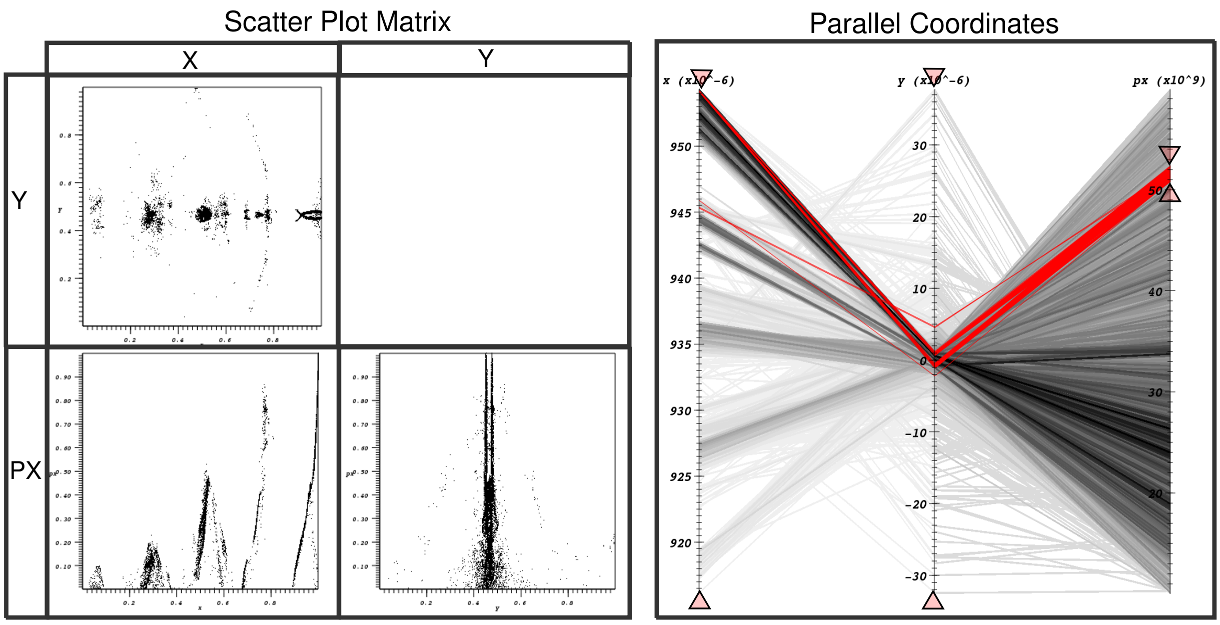


Figure 2.4: Left: Scatter plot matrix of an example subset of a particle accelerator simulation dataset. Right: The same data subset visualized with parallel coordinates. Two sliders are added to each parallel axis which are used to define multi-dimensional range queries. In this example a lower and upper threshold is defined in *px* (rightmost axis). The selected data subset (the focus) is rendered on top of the main plot (the context).

range of interest in each displayed data dimension. By rendering selected data subsets (the focus) on top of the main plot of the complete data (the context), a user receives immediate feedback about important properties of a selection such as the main trends, clusters, and outliers. A simple example for data selection in parallel coordinates is shown in Figure 2.4 (right).

Parallel coordinates also have substantial disadvantages. Parallel coordinates are an expensive plot in the sense that many pixels are required to represent a single data record. The overplotting of the many polylines required to display all data records from a large dataset quickly causes the plot to appear cluttered and leads to data occlusion. In particular in the context of large datasets, connections of polylines at the different axes can become visually misleading so that it is often impossible to follow the trend of a single data record through the plot. Parallel coordinates are also an order dependent visualization, i.e., depending on how the different axes — each representing one data dimension — are ordered, a user may be able to see different relationships in the data.

Many extensions of parallel coordinates focus on improving the rendering, and in this way the visual appearance and interpretability of parallel coordinates. Color is commonly used to ease distinction of individual data lines as well as to represent additional information to allow,

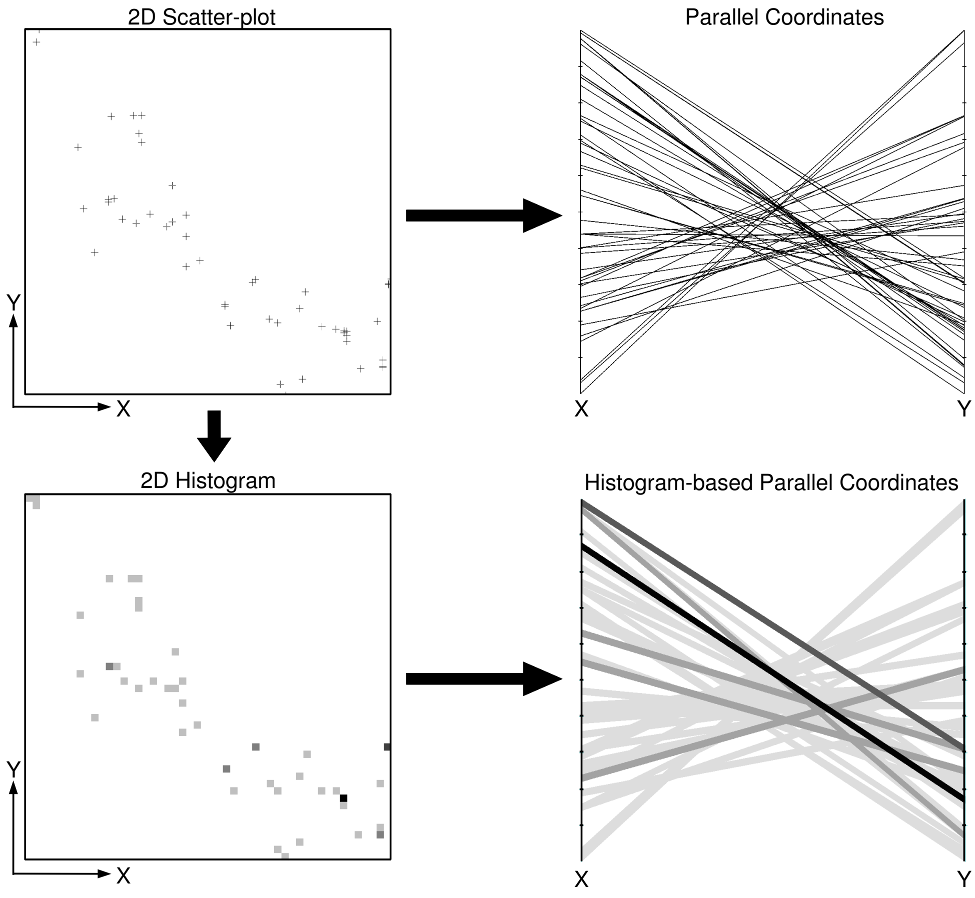


Figure 2.5: Illustration of line-based parallel coordinates (top) and histogram-based (binned) parallel coordinates (bottom).

e.g., detection of relationships between one variable and all other data dimensions displayed. Wegman and Luor [16] proposed the usage of transparency and over-plotting of translucent data points/lines. This method highlights dense areas while sparse areas fade away, thus revealing inherent data characteristics. With no *a priori* knowledge about how many data lines pass through the same pixel, the alpha channel of color becomes quickly saturated so that the effectiveness of this approach is limited in particular in the context of large data. Graham et al. proposed the use of curves instead of poly-lines to improve the visual display and readability of parallel coordinates [17]. Moustafa and Wegman proposed generalized parallel coordinates, a generalization of parallel coordinates based on parameter transformation [18]. Depending on the applied interpolation function regular parallel coordinates (using piecewise Lagrangian interpolation), smooth parallel coordinates (using spline-based interpolation), and other variants of parallel coordinates can be created.

Other extensions to standard parallel coordinates focus on the parallel axes themselves and their generalization. Re-ordering [19], scaling, and flipping of parallel axes as well as data reduction [20] have been used to reduce visual clutter and to enable detection of different relationships in high-dimensional data. In order to display additional information within parallel coordinates, a common approach is to generalize the parallel axes to be plots themselves. Commonly the parallel axes are simply overlaid with box-plots or histogram plots of the data to provide additional information about statistical properties of the data [21]. Parallel coordinates have also been extended to three-dimensional (3D) visualizations by modifying the parallel axes to be two-dimensional (2D) plots. Using this principle, Fanea et al. [22] combined star-glyphs and parallel coordinates in which each axis of a parallel coordinates plot is itself a star-glyph. Rübel et al. [157] combined 2D scatter plots and parallel coordinates to enable effective visualization of 3D gene expression data via 3D parallel coordinates (see Section 4.2.2.3 for details).

Several extension to standard parallel coordinates focus on the representation of data subsets. Effective display of data subsets is particularly important in the case that parallel coordinates are used for data mining through manual data selection as well as analysis of pre-classified data. Distortion operations, such as dimensional zooming, support a more detailed analysis of data subspaces [23]. Fua et al. [23] and Novotny [24] proposed the usage of color bands for visualization of selections (or clusters) in parallel coordinate views and McDonnell et al. proposed illustrative parallel coordinates.

Many extensions to parallel coordinates have been proposed with the goal to enable visual analysis of large datasets by creating data-dependent aggregate visualizations. Fua et al. [23] proposed using hierarchical parallel coordinates based on hierarchical clustering to create a multi-resolution view of the data that enables data exploration at varying levels of detail. Johansson et al. [25] used clustering to determine the inherent structure of data and displayed that structure with high-precision textures using different texture transfer functions. Novotný used a binning algorithm based on a *k-means* clustering approach for creating an aggregate parallel coordinates visualization [24]. All these approaches are well-suited for presenting static data, but are not well-suited for time-varying data since defining temporally consistent clusters is non-trivial and computationally expensive.

The histogram-based parallel coordinates approach proposed in this thesis (see Section 5.4.2) extends the work of Novotný and Hauser [26], who proposed using binned parallel coordinates as an output-oriented (rendering) approach. Rather than rendering one line per data point, the relationship between neighboring axes is first discretized via 2D histograms (see Figure 2.5). Each non-empty bin of a 2D histogram is represented by a rectangle connecting the corresponding data ranges on the respective neighboring axes. The main limitation of the approach proposed by Novotný and Hauser is that in order to achieve interactive rendering speed, their method pre-computes all possible 2D histograms with a fixed resolution of 256×256 using regular, equal-sized histogram bins. To implement different level-of-detail views, bins in the pre-computed histograms are merged, reducing the number of bins by half in each drill-down step. While this strategy is efficient, it has several limitations. First, it does not allow for smoothly drilling into finer-resolution views of the data. Second, it supports presentation of binned views of the entire dataset, but does not support user-defined data subsetting. Finally, the fixed 256×256 histogram resolution exhibits significant aliasing when zooming in on narrow variable ranges in the parallel coordinates plot. Their approach uses binned parallel coordinates for context views and traditional, polyline-based parallel coordinates for focus views. These focus views may still contain a substantial number of data records and suffer from extensive occlusion as a result.

2.2.3 Descriptive Statistic Plots

Descriptive statistics are used to quantitatively describe the main features of a data collection. By summarizing data distributions through a small set of parameters, important data characteristics can be extracted and represented in a concise fashion. In the context of this research, descriptive statistic are used mainly for the purpose of understanding data subset characteristics essential for validation of results from data classification and data mining. Common measurements used in the context of descriptive statistics are the median, mode, mean, variance, standard deviations, and quantiles. The most common statistics plots used in the context of this work are the box plot, histogram plot, average curve plot, and scatter plot. Scatter plots were introduced in Section 2.2.1. This section focuses on the box, average curve, and histogram plot. A detailed introduction to statistics and their use for data analysis is provided in [27].

The box plot is a standard technique for presenting summary statistics. In the context of this work, the box plot is used to show minimum and maximum range values, standard deviation, and average value. Depending on the application, other values, such as upper and lower quartiles instead of standard deviation or median instead of mean, may be shown. Figure 2.6a illustrates the typical box plot variant used in this work. The box plot is bound by two lines indicating the minimum and maximum value of the data and an additional line showing the average value. A box centered around the average value is used to indicate the standard deviation. The box plot provides and easy way to summarize the distribution of a dataset through the display of a collection of statistical values. Furthermore, the reduced box-plot representation allows for fast comparison of different datasets (here often subsets of the same dataset), since only a few

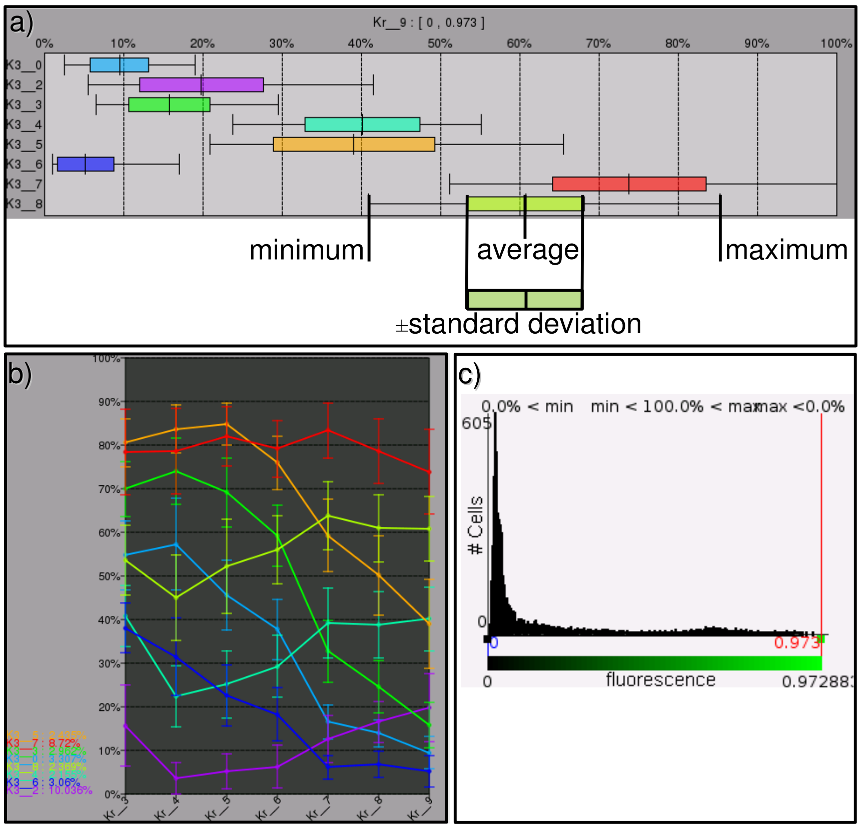


Figure 2.6: Example statistic plots created using the visualization system PointCloudXplore (see Section 4.2). a) Box-plot comparing the expression characteristics of a set of clusters (each selecting a subset of cells) in a selected gene. b) Average curve plot with error bars showing the average expression profiles of a set of clusters in a series of genes. c) Histogram plot of the expression values of a single gene in all cells of a *Drosophila* embryo.

statistical measurements, rather than the whole data, need to be analyzed. A survey of the box plot is provided by Potter [28].

Average curve plots are commonly used for visualization of data trends (see Figure 2.6b). Similar to parallel coordinates, the different data dimensions are arranged along the horizontal axis (x). The vertical axis (y) is used for display of average values in the different data dimensions. Neighboring average values are connected via line segments to indicate data trends. Average curve plots are often used to show, e.g., the variation of variables over time. Depending on the distribution of the data the average value may not be a good representative value. Error bars indicating, e.g., the standard deviation, are therefore often added to the plot. In some cases the average curve plot is also combined with the box plot.

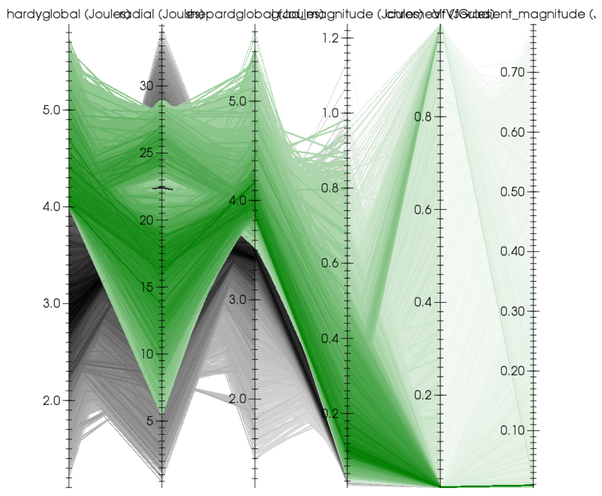
Histograms are commonly used for investigation of data distributions. A histogram plot is a graphical way of presenting a frequency distribution. The data (if not categorical) is first subdivided into a set of bins, each representing a discrete data range. The value of each bin is defined based on the number of data records within the corresponding data range. Histograms are commonly visualized using bar-graph or curve plots (see Figure 2.6c).

2.3 Linking Scientific Visualization and Information Visualization

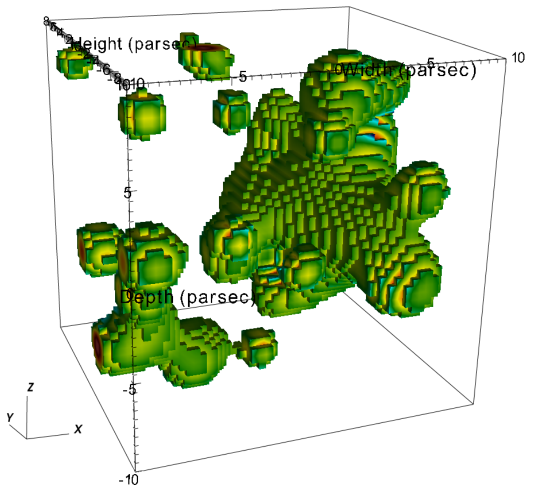
Multiple views, each highlighting a different aspect of the data, are commonly used to analyze complex data. In the simplest form, multiple visualizations of various variables of the same data are visualized in multiple views with the same physical context, e.g., one map for temperature and a second map for pressure. Linking of the camera view and timestep allows for effective linking of different scientific visualizations with the same physical context.

With scientific data becoming increasingly high-dimensional and large, systems that support linking of multiple different views become increasingly important. Baldonado et al. described in their work a set of guidelines for multiple view systems [29]. Henze [30] proposed a system based on multiple views (termed *portraits*) for exploration of time-varying computational fluid dynamics data sets; advanced queries can be performed by selecting data subsets in these portraits. In the WEAVE system, a combination of Physical Views and Information Visualization Views is used for exploration of cardiac simulation and measurement data [31]. Doleisch et al. [32] formalized the concept of using abstract views to define data queries, a concept that is implemented, e.g., in the SimVis system [33].

It is often useful to interactively select data samples from a visual data representation, an operation generally referred to as *brushing*. A *brush* is an object that defines one specific selection of data samples. Furthermore, by highlighting brushes defined in one view in all other views, identification of further data properties becomes possible. In the literature this process is also termed *linking*. Linking multiple views via the concept of brushing enables detailed analysis of high-dimensional data. To enable definition of data selections based on the information shown in different views, one may combine multiple brushes via logical operations such as AND (intersection), OR (combination), or NOT (inversion) [32]. Interactivity is important in the context of data selection for visual data exploration. Providing immediate feedback about selection results enables iterative definition and refinement of data queries and supports accurate feature selection. Query-driven visualization addresses this issue by combining visualization and state-of-the-art index/query methods (see Section 2.5).



(a)



(b)

Figure 2.7: a) Parallel coordinates plot of the complete dataset (gray) and a user-defined selection (green). b) Pseudocolor plot of the same user-defined data subset as shown in *a*. The visualizations were created using the VisIt system (see Section 2.4) and show the VisIt example dataset *noise.silo*.

In Figure 2.7, parallel coordinates are used to select cells with high values in the variable *hardyglobal* of VisIt's example dataset *noise.silo* (see also Figure 2.1). While parallel coordinates are an effective tool for display and selection in high-dimensional data, the relationships between different cells in physical space are not intuitively revealed. A pseudocolor plot is, therefore, linked to the parallel coordinates view to evaluate the spatial patterns defined by the selection.

2.4 High-performance Visual Data Analysis using VisIt

Many commercial and open source visualization packages are available for visualization of large scientific data, such as VisIt [34, 35], ParaView [36], or EnSight [37]. This section focuses on VisIt. VisIt is widely used within the DOE[1] research community and is used in this work for visual analysis of laser wakefield particle accelerator simulation data. VisIt was originally described by Childs et al. [35] who demonstrated a data processing and visualization architecture that is capable of scaling to extreme dataset sizes. VisIt [34] has been shown to scale to several billion data points per timestep and runs in parallel on nearly all modern HPC platforms.

[1]*DOE=Untied States Department of Energy.* See `http://www.energy.gov/`

The basic design of VisIt follows the client-server model with the most expensive computations being executed in parallel by the server (see Section 2.4.1 for details). The client-server concept supports effective remote visualization while the parallel implementation of the server enables fast processing of extremely large datasets. VisIt's remote visualization capability crosses several independent axes. In addition to parallelization of data fetching, VisIt also performs data extraction and calculation entirely in parallel. Finally, rendering may be done in serial or parallel, depending on the data load. If the resulting geometry is small enough, it is collected at the HPC side and transferred across the network to the user's desktop for GPU-based rendering. However, if the geometry is too large for interactive display on a single GPU, VisIt employs sort-last rendering and compositing on the HPC system and transferres the computed pixel information (i.e., the image) across the network to the user's display. This so-called *scalable rendering mode* model has many parallels to remote visualization architectures such as those in ParaView and EnSight.

VisIt employs a contract-based data flow network to enable efficient visualization while allowing new components to be flexibly added via plugins (see Section 2.4.2 for details). Via the contract, each component can describe its impact on the pipeline allowing dedicated optimizations to be applied. VisIt's plugin capability allows for dynamic incorporation of new plots for data display, operators for data manipulation, or database modules for data access. VisIt currently supports over twenty plots, more than forty operators, and over eighty file formats. VisIt furthermore supports analysis concepts such as expressions (i.e., creation of derived quantities) or queries (i.e., extraction of quantitative information from data). The following two subsections provide an overview of the high-level design of VisIt (see Section 2.4.1) and the contract-based visualization concept it employs (see Section 2.4.2).

2.4.1 High-level Design of VisIt

Figure 2.8 depicts the high-level design of VisIt as described in [38] and [39]. VisIt consists of several main components (essentially different programs) that communicate with each other through sockets. The main components running on the client side — usually the user's local workstation — are the (i) *GUI*, (ii) *Viewer*, and (iii) Command line interface (*CLI*). The main components running on the server side — either a remote computer or the user's workstation — are the (iv) VisIt component launcher (*VCL*), (v) database or meta-data server (*mdversver*), and (vi) the compute engine (*Engine*) running in serial (*engine_ser*) or parallel mode (*engine_par*).

The main purpose of the VisIt components running on the local workstation is to provide appropriate user controls and to display analysis results. The main user controls are provided via

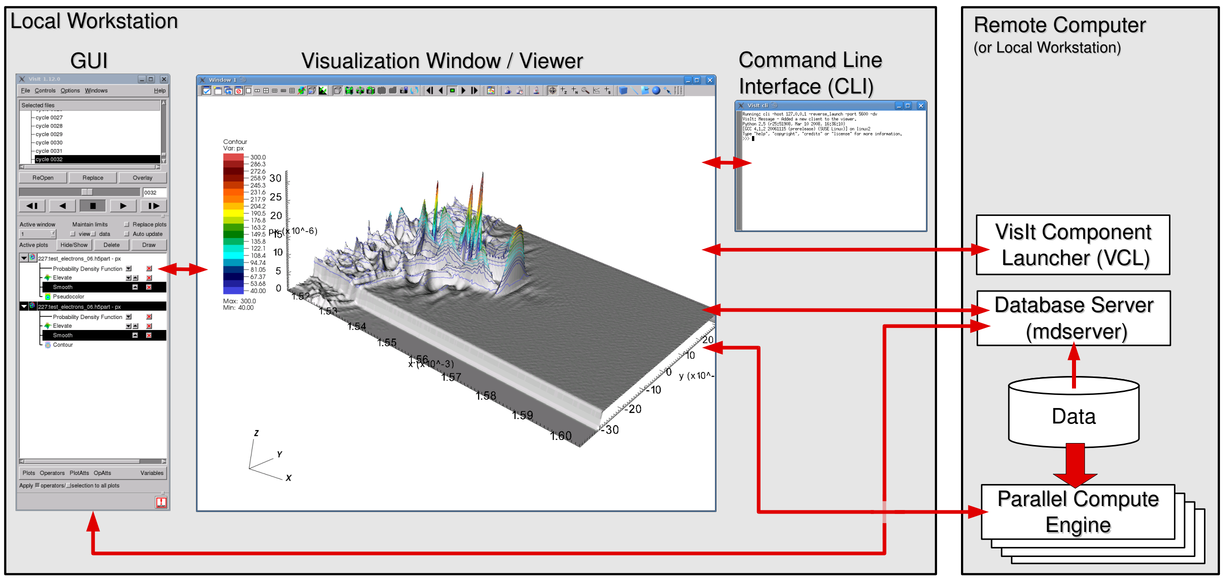

Local Workstation
GUI
Visualization Window / Viewer
Command Line Interface (CLI)
Remote Computer
(or Local Workstation)
VisIt Component Launcher (VCL)
Database Server (mdserver)
Data
Parallel Compute Engine

Figure 2.8: High-level design of VisIt.

the GUI and the CLI interface. The GUI provides a graphical interface allowing the user, e.g., to load data, set up visualization pipelines, define expressions and queries, or to define system settings. The CLI interface provides a python-based command line interface for the control of VisIt. The CLI is useful, e.g., to define macros or to implement visualization scripts for plots and animations. The Viewer component serves two main purposes. VisIt's state is centralized in the viewer, i.e., when VisIt's state changes, then the Viewer notifies the other compontents and updates the system accordingly. The Viewer is furthermore responsible for managing visualization windows including rendering, display, and manipulation of visualizations.

The VisIt components running on the server side are responsible for performing the actual analysis including data loading, execution of the analysis and visualization, and if requested also rendering. The VisIt Component Launcher (VCL) is a small component launched by VisIt on remote computers when running in distributed mode. The main purpose of the VCL is to launch jobs on remote computers. The mdserver is a lightweight component for file access. The mdserver browses remote file systems to create listings of the contents of remote directories and loads meta-data of files to allow the user to set up visualization pipelines without an active compute engine. The Engine is the component that performs the actual data processing in response to requests from the Viewer. VisIt supports a serial (engine_ser) as well as a parallel compute engine (engine_par). In case that the user requests the scalable rendering mode, the Engine also performs the rendering and sends pixels (rather than geometry, i.e., polygons, lines, and/or points) to the

Viewer for display.

The communication between the different components of VisIt is implemented on the lowest level via sockets. To ease communication, two additional communication layers are built on top of the low-level, socket-based communication. The first communication layer is designed for exporting the system state. The Viewer manages the system state in various *AttributeSubjects* objects. User interface modules subscribe to a state, so that the state is automatically pushed to the subscribed modules upon change. The second high-level communication component implements remote procedure calls (RPCs). RPCs are implemented via proxy classes that other modules can link to and issue method calls which in turn become RPCs. RPCs are used when one component needs another component to perform a particular operation. Within VisIt, RPCs are used to allow, e.g., the CLI or GUI to initiate changes of state in the Viewer, to allow the Viewer to request that the mdserver should open a file, or to allow the Viewer to initiate an action in the Engine.

2.4.2 Contract-based Visualization Pipeline

VisIt employs a contract-based data flow network design that allows for pipelining and I/O optimization, reducing unnecessary processing and disk access [35]. The two base object types of the network are *data objects* and *components*. There are three types of components, which are *filters*, *sources*, and *sinks*. Sources are the origin of the data and, hence, only have an output which are data objects. Filters take data objects as input, apply some transformation, and output the result again as data objects. Sinks have only data object inputs. A *pipeline* is defined by a collection of components with a source component at the beginning, followed by a series of filter components, and finally a sink component. Figure 2.9, illustrates a typical execution pipeline as described in [35, 38]. The source is typically a file reader responsible for reading data from disk and the sink component is typically a rendering engine. When run in parallel, VisIt sets up identical data flow networks on every processor, each of which is responsible for a portion of the data. The *load balancer* decides in this case which portion of the data each processor should process.

The execution of a VisIt pipeline is demand-driven. The first step of a pipeline update consists of a pull operation by the sink to acquire the needed data. The pull operation in turn causes the sink to generate an update request that is propagated up the pipeline via the filter components to the source. The source component generates (loads) the requested data. Execute phases afterwards propagate through the pipeline until the sink is reached, i.e, the data object created by the source serves as input to the first filter which performs some operation and passes the resulting

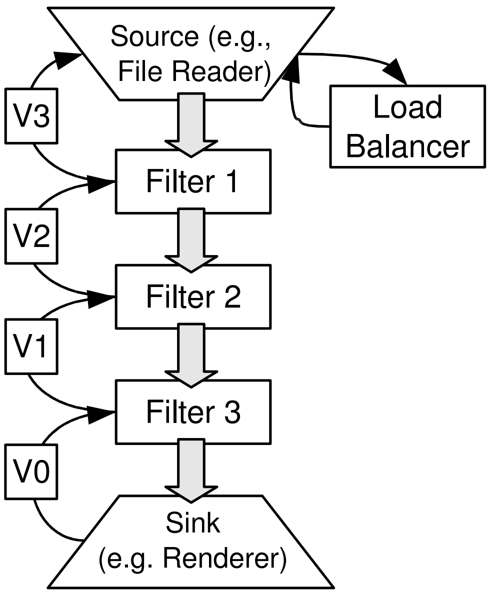


Figure 2.9: Illustration of a contract-based visualization pipeline. The thin arrows denote the update of the pipeline. During the update, the original version of the contract (*V0*) originates from the sink, which is often a renderer. The contract is passed through the different filters of the pipeline, each of which may modify the contract according to its specific needs (contract version *V1*-*V3*). The load balancer decides on the further execution and passes the information to the sink (usually a file reader). The sink loads the data and initiates the execute phase of the pipeline (thick arrows in the figure).

output data object on to the next filter. In this manner the data is passed from filter to filter (each performing some operation) until the sink component is reached.

In VisIt, a *contract* is passed through the pipeline during the update phase along with the update request. The contract allows different components of a pipeline to communicate with each other. Using the contract, each component can describe its impact on the pipeline, allowing dedicated optimizations to be dynamically applied to, e.g., minimize the data read from disk, modify how the data is treated, or to modify the way the data moves through the pipeline. The contract provides an abstraction from the specification of components allowing description of the impact of different components without specific knowledge about the individual components themselves. This contract-based design eases (i) integration of new components into VisIt (a new component simply needs to be able to describe its impact) and (ii) management of a large number of components, because different components can be developed and maintained independently of each other.

The different main components of the pipeline (plots, operators, databases) are implemented in VisIt as plugins. A plugin is a library that is loaded at startup. The plugin concept allows developers to add new capabilities to VisIt without having to modify the actual core system.

Using VisIt's extensible contract-based system, this work implements efficient methods for particle selection and particle tracing in VisIt and presents a novel approach for rendering of histogram-based parallel coordinates used for efficient data exploration (see Section 5.4 for details).

The following section introduces query-driven visualization (QDV) (see Section 2.5), a feature-based visualization concept used in the context of this work to enable efficient exploration of extremely large data sets. Afterwards, Section 2.6 introduces FastBit, a data index/query system used in this work for implementation of QDV within VisIt.

2.5 High-performance Query-driven Visualization

The term *query-driven visualization* (QDV) was introduced by Stockinger et al. to describe a combination of high-performance indexing and querying capability (see Section 2.6) with visual data exploration tools [40]. Aimed towards the analysis of massive datasets, QDV allows users to quickly search the data for features of interest. The visualization is focused on the selected features, significantly reducing the amount of data presented to the user.

In their work, Stockinger et al. [40] compared the performance of (i) their index/query system based on bitmap indexing combined with visual data exploration with (ii) state-of-the-art, tree-based searching structures that form the basis for a widely used contouring implementation. Their work shows that their approach outperforms tree-based search structures for scalar variables, and also points out that all tree-based index/search structures are not practical for large, multivariate datasets since they suffer from the so-called *curse of dimensionality* [41]. The basic idea is that storage complexity grows exponentially as one adds more and more search dimensions (e.g., more variables to be indexed/searched).

These concepts were later extended to the analysis of massive collections of network traffic data in two related works. First, the notion of performing network traffic analysis using statistics (e.g., histograms) rather than raw data led to a methodology that enabled exploration and data mining at unprecedented speed [42]. That study showed the use of these concepts to rapidly detect a distributed scan attack on a dataset of unprecedented size – 2.5 billion records. Users are presented with an interface consisting of histograms of individual variables, and then they formulate a complex query via a process that is essentially a histogram "cross product." The process of data mining was subsequently accelerated through a family of algorithms for computing conditional histograms on SMP parallel machines [43].

Visualization techniques, like QDV, that focus on specific features of the data are also referred to as *feature-based visualizations*. In the context of QDV, the features of interest are usually defined by the user via dedicated data queries. Other featured-based visualization methods automate the feature detection step by choosing an appropriate filtering process [6].

As part of this research effort, the extension of the concept of QDV to the analysis of laser wakefield particle acceleration (LWFA) data is described. By combining the state-of-the-art index/query system FastBit with high-performance visualization using VisIt, efficient exploration of extremely large, complex, high-dimensional data becomes possible (see Section 5.4). This thesis also describes novel methods for automatic feature detection in LWFA simulation data and presents dedicated visualizations for the analysis of the so defined features (here particle bunches). To achieve good performance in the context of QDV as well as automatic data analysis, this work makes use of efficient data access methods described in the following section.

2.6 High-performance Index/Query for Data Mining

In many cases only a subset of the data is actually relevant for the data analysis. In the context of laser wakefield particle acceleration, e.g., only a fraction of all particles is accelerated to relevant levels of energy and are of interest for the analysis. In the context of query-driven visualization, e.g., a user usually issues a series of queries, each focusing on different features of interest or refining earlier queries. Automatic analysis algorithms often employ dedicated data reduction methods to ensure that at each analysis step only the relevant data is considered.

To achieve good performance it is paramount that the visualization and analysis algorithms are able to identify and access data subsets of interest quickly. This is particularly important in the context of extremely large data sets. In the context of the analysis of extremely large LWFA simulations, this work makes use of state-of-the-art data management to be able to quickly: (i) evaluate range queries, (ii) compute conditional histograms, and (iii) trace particles using ID queries, significantly improving the overall computational performance of the proposed analysis.

The commonly used strategy for accelerating selective data accesses is called *indexing* in database terminology. A standard database indexing technique is the B-tree [44]. B-trees have properties favorable for transaction-based applications that require frequent updates to the underlying base data and the index. The data access patterns for data mining and analysis applications, however, tend to be read-only. The database indexing technology that is best suited for this type of data access is known as the bitmap index [45, 46]. The core idea of a bitmap index is to use a

RID	1	2	3	4	5	6	7	8
I	0	1	3	2	3	3	1	3
bitmap index								
=0 (b_1)	1	0	0	0	0	0	0	0
=1 (b_2)	0	1	0	0	0	0	1	0
=2 (b_3)	0	0	0	1	0	0	0	0
=3 (b_3)	0	0	1	0	1	1	0	1

Figure 2.10: A sample bitmap index where RID is the record identifier and **I** is the integer attribute with values in the range of 0 to 3. Each bitmap b_i represents one value j. Each bit of a simple bitmap b_i represents one data record and indicates whether the value of the corresponding record is j (bitvalue 1) or not (bitvalue 0). Using bitmap indexing, one only needs to perform bitwise logical operations such as $b_3 \mid b_4$ to find the records that satisfy conditions such as $\mathbf{I} > 1$.

sequence of bits to mark the positions of records satisfying certain conditions.

The illustration in Figure 2.10 shows four simple uncompressed bitmaps, b_1, b_2, b_3, and b_4, where each of them represents whether the value of **I** is 0, 1, 2, or 3. To find the rows that satisfy condition such as $\mathbf{I} > 1$, one only needs to perform bitwise logical operations such as $b_3 \mid b_4$. Because bitwise logical operations are well-supported by all computers, the answer to such conditions can be computed efficiently. Based on a bitmap index, the data records that correspond to a query, such as $X > 5$, can be identified without having to access the raw data [47, 48].

Without compression, the size of a bitmap index increases linearly with the number of bitmaps used. In uncompressed form, such bitmap indices may require too much space for variables with many distinct values, such as particle position or momentum. To improve the efficiency of the bitmap indices on such variables and control index size, many different techniques have been proposed [49, 50]; the most common ones being *binning*, *encoding*, and *compression*. Binning and encoding are different ways of controlling the number of bitmaps per index. Compression is used to reduce the size of individual bitmaps.

To enable efficient analysis of extremely large LWFA simulation data, this research makes use of a bitmap index software called FastBit [51]. FastBit implements the fastest known bitmap compression technique [52, 53], and has been demonstrated to be effective in a number of data analysis applications [42, 54]. In particular, it has a number of efficient functions for computing conditional histograms [43], which are crucial for this work. Furthermore, FastBit indices are relatively small compared to popular indices such as B-trees [52, Fig. 7] and can be constructed much faster than others [47, Fig. 12]. Bitmap indices are well-known for their effectiveness on data with a relatively small number of distinct values, such as gender. FastBit indices have been demonstrated to be very efficient also for data with a large number of distinct values through its unique compression [50] and binning [48] capabilites.

When a variable has a large number of distinct values, such as temperature expressed as floating-point values, the corresponding FastBit index is typically based on a binned version of the data, where a bit is 1 if the value of a record falls in a particular bin. In this case, the number of 1s in a bitmap corresponding to a bin is the number of records in the bin. This provides an efficient method for computing a histogram [43]. FastBit offers a number of different options for creating bitmap bins. When composing range queries, users typically specify conditions with relatively low-precision values, such as pressure less than $1 * 10^{-5}$ or momentum greater than $2.5 * 10^{8}$. The constant $1 * 10^{-5}$ is said to have 1-digit precision, and the constant $2.5 * 10^{8}$ have 2-digit precision. FastBit can build indices with bin boundaries with any user-specified precision so that all queries involving low-precision boundaries can be answered accurately based on the index only.

The work presented in Chapter 5 makes extensive use of two additional features of FastBit specifically enhanced for this work: i) ID queries; and ii) a special function for computing of 3D conditional histograms. To track a set of particles in a LWFA simulation dataset through time, FastBit can extract all particles with a given set of identifiers (IDs) from all time steps via queries of the form $ID = 0 || ID = 1 || \ldots || ID = n$. This query may involve several million IDs, which can take a long time just to parse the query string. FastBit provides a mechanism to directly input the list of IDs to reduce the query response time.

FastBit has shown to be effective for the computation of 1D and 2D conditional histograms (see [43] and [158]). This research extends FastBit to enable efficient computation of 3D conditional histograms and to export bit vectors for representing particles (data records) associated with the bins of a histogram. These bit vectors allow the analysis to directly retrieve the information related to the corresponding particles enabling fast data-access, leading to significant improvement of the computational performance of the automatic beam path analysis pipeline described in Section 5.6.

2.7 Data Classification for Scientific Data Analysis

Data classification is the systematic grouping of data into categories according to some criteria. Data classification methods are commonly used for identification of data subclasses and discovery of characteristic features of the underlying data as well as for simplification of the data representation to facilitate data analysis.

Data classification may be performed using automatic, semi-automatic, or manual classification procedures. One sometimes also distinguishes between segmentation and machine-learning-based classification methods. Segmentation methods commonly subdivide the data based on predefined criteria, e.g., by identifying regions associated with maxima of the data, whereas machine learning methods typically try to minimize an objective function. In practice, such a clear distinction is, however, often not possible. In the context of machine learning, one commonly distinguishes between supervised and unsupervised learning methods. Supervised learning refers to a set of techniques for learning of a function from training data. Supervised learning may be used for prediction of continuous functions (i.e., regression) or to predict class labels (i.e., classification). The training data for supervised learning consists in general of pairs of input data objects and the desired output. In contrast to supervised learning, unsupervised learning methods seek to predict a model directly from unlabeled data, i.e, without the use of pre-labeled training data.

This research makes use of two main data classification methods, i.e., *region-growing* and *data clustering*. Region-growing refers to a set of methods for segmentation of data into subgroups by identifying regions that correspond to a set of seed points (see Section 2.7.1). Data clustering is a class of techniques for unsupervised classification of data samples into groups of similar behavior (see Section 2.7.2). In the context of this work, region-growing is commonly used for detection of specific data features (i.e., identification of a set of data samples from the complete data), whereas data clustering is used for partitioning of the complete data into subclasses. This section concludes with a description of the concept of *ensembles*. Ensembles are a set of methods aimed at integration of multiple learned models (see Section 2.7.3).

2.7.1 Introduction to Region-Growing

Region-growing refers to a set of algorithms that group subregions into larger regions based on predefined criteria for growth. Region-growing is a popular approach in image analysis and is here used for image segmentation [55, 56, 57]. The basic method is to start with a set of seed points. From the seed points, the algorithms grows larger regions by appending to each seed those neighboring pixels with predefined properties similar to the seed (such as specific ranges of the variables) [55]. Depending on the application, seed points for region-growing may be defined automatically (e.g., as maxima of the underlying function) or manually by the user. Commonly, a set of seeds is selected based on properties of the underlying data. In case that no *a priori* information about good seeds is available some methods use as many starting points as the total number of data points. The procedure is to compute the same set of properties for each starting point to allow region-growing. Besides selection of proper seeds, criteria such as

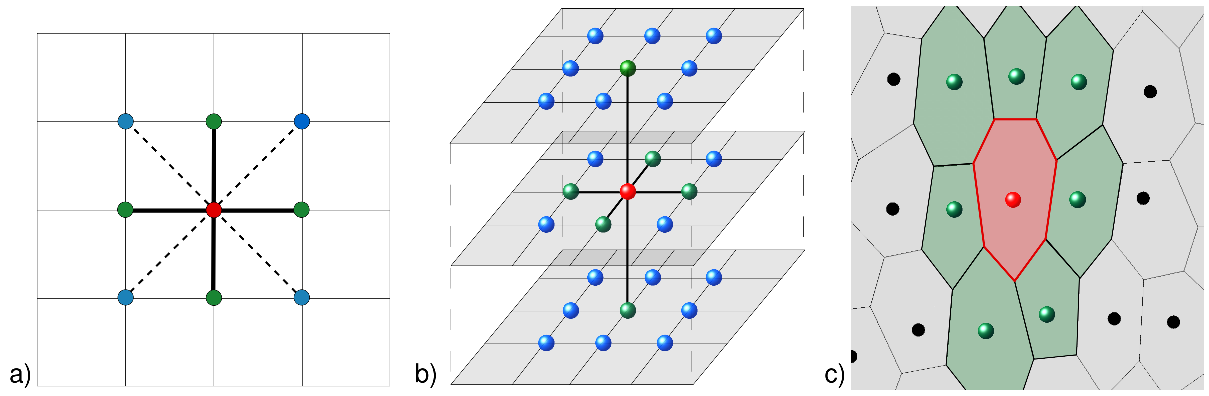


Figure 2.11: Illustration of common topologies used to define the neighbors of data points. In all figures the center point is shown in red, its direct neighbors in green, and its diagonal neighbors in blue. a) Common neighbor stencils in 2D image data include the (i) 4-neighbor stencil (only green neighbors) and the (ii) 8-neighbor stencil (green and blue neighbors). b) Common neighbor stencils in 3D image data include the (iii) 6-neighbor stencil (only green neighbors) and the (iv) 26-neighbor stencil (green and blue neighbors). c) Illustration of the topology used in the case of 3D gene expression data. Cells that share an edge in the approximated Voronoi tessellation of the embryo's blastoderm surface are considered neighbors.

the used topology (neighborhood) and stop criteria are important parameters of region-growing algorithms.

The used topology determines which points are considered as neighbors of a data point and, hence, defines the possible directions of growth. In the context of image segmentation, pixels are typically organized on a regular grid. Depending on which pixels are considered as neighbors, different types of neighborhood stencils can be defined, e.g., in the case of 2D image data the (i) 4-neighbor stencil, i.e., only direct neighbors in positive and negative x and y direction, or the (ii) 8-neighbor stencil, i.e., the 4-neighbor stencil plus the diagonal pixel neighbors (see Figure 2.11a). Similarly, the 6- and 26-neighbor stencil can be defined in the case of 3D image data (see Figure 2.11b). In more complex cases, such as scattered data, a topology may not be given and must be defined first. In the case of 3D gene expression data, the neighborhood of a cell is defined based on the approximated Voronoi diagram of the embryo's blastoderm surface [159] (see Figure 2.11c).

The proper choice of stop criteria largely depends of the underlying problem. In the simplest case, stop criteria are defined via thresholds in the underlying scalar field, either manually by the user or automatically by the algorithm, e.g., relative to the scalar values of the seed. When derivatives of the underlying scalar field can be defined, other stop criteria, such as "Stop the growth when an inflection point of the function is reached," may be employed.

Region-growing is used in the context of 3D gene expression data for automatic seed cell selection [159] (see Section 4.2.2.2). This research also employs region-growing methods based on three-dimensional histograms of the data for identification of particle bunches in laser wakefield particle accelerator simulation data (see Section 5.6.4).

2.7.2 Introduction to Data Clustering

Data clustering is a class of techniques for unsupervised classification of data samples (e.g., cells) into groups (clusters) of similar behavior. Data clustering provides a means for the automatic discovery of data subclasses [58].

With increasing size and complexity of today's scientific data, computational methods, such as data clustering, that promise to enable automatic detection of characteristic data features become increasingly important. Researchers are faced with ever increasing amounts of data and information, prohibiting a purely manual analysis process. Data clustering is a challenging problem. Depending on the underlying data, clusters may have significantly different structures. Besides few generic concepts and methodologies, this has led to the development of a large range of clustering algorithms specifically designed for the analsis of specific types of data.

A taxonomy of clustering algorithms can be defined based on various criteria. The most common criteria are based on the type of output a clustering algorithm produces (hard/fuzzy-partition and hard/fuzzy-hierarchy), the type of criteria employed to define clusters (e.g., density-based, model-based, or mean-square-error), as well as the type of algorithms employed to compute the clustering and their various computational characteristics (e.g., graph-based, expectation maximization, artificial neural network (ANN), or agglomerative vs. divisive and stochastic vs. deterministic). Based on the output of clustering algorithms, one commonly distinguishes between hard and fuzzy clustering algorithms. A hard clustering algorithm assigns each data item to exactly one cluster, whereas a fuzzy clustering algorithm defines for each data item a vector describing its degree of membership in all clusters, i.e., the likelyhood of the data item to belong to a particular cluster. A hard clustering can be derived from a fuzzy clustering simply by assigning each data item to the cluster that it is most likely to belong to. Other common criteria for differentiation of clustering algorithms include, e.g., deterministic vs. stochastic algorithms and partitioning vs. hierarchical clustering algorithms. Figure 2.12 shows an example taxonomy including all clustering algorithms used in the context of this thesis. The shown hierarchy of clustering algorithms is a refinement of the taxonomy proposed by Jain et al. [58].

In order to be able to identify groups of similar data items, the distance between data items needs to be defined. Depending on the data to be clustered, different distance metrics may

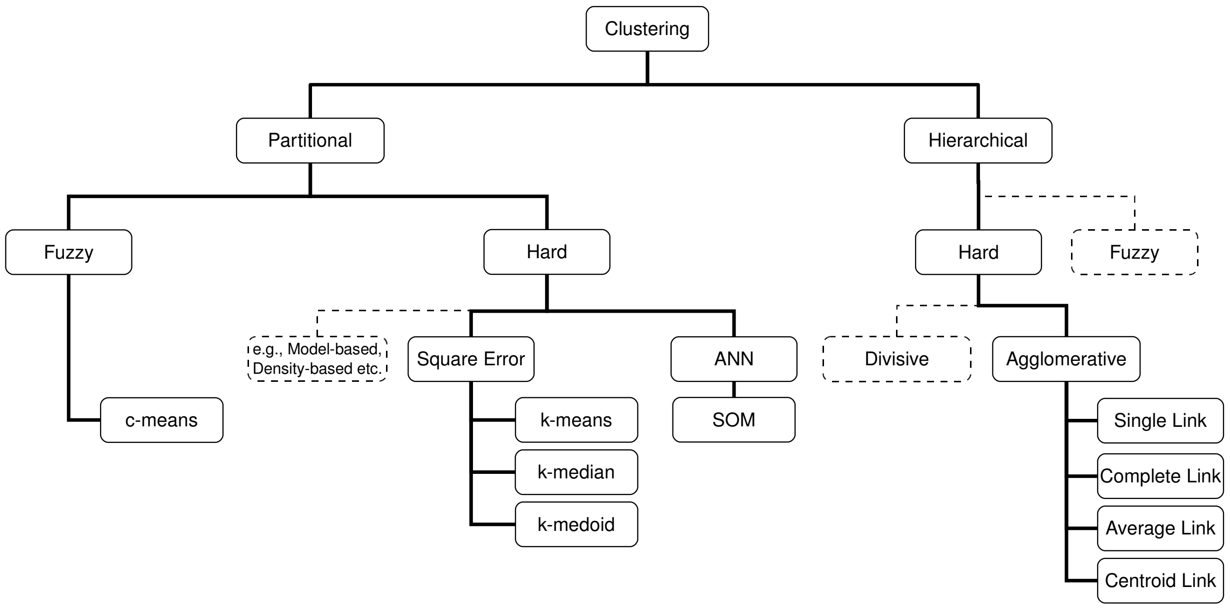


Figure 2.12: Taxonomy of the clustering algorithms used in the context of this thesis. Dashed lines indicate types of clustering algorithms not employed in the context of this research shown here only for the purpose of completeness.

be more suited than others. Common distance functions used in the context of clustering of scientific data include the Euclidean distance, City-block distance, as well as various derivatives of the Pearson correlation coefficient. A detailed overview of the distance functions used in the context of this research for clustering of 3D gene expression data is provided in [160] as well as the manual of the clustering library *Cluster 3.0* [59].

The next section first provides an overview of the most common clustering algorithms used in this research effort (see Section 2.7.2.1). Section 2.7.2.2 afterwards provides an overview of computational methods for cluster validation and estimation of the number of clusters k. This introduction to data clustering concludes with a brief overview of applications of data clustering to gene expression data (see Section 2.7.2.3). Having introduced various data classification algorithms that produce a single classification of the data (e.g., region-growing and clustering), this chapter concludes with a description of ensembles, i.e., methods for integration of multiple learned models (see Section 2.7.3).

2.7.2.1 Clustering Algorithms

Partitioning algorithms classify the data into a set of k clusters. In the case of clustering algorithms like k-means, k-medians, and k-medoids, k is a user-defined input parameter. Other methods, such as mean shift clustering algorithms, determine the number of clusters automatically during clustering [60, 61]. Self-organizing maps (SOMs) form a nonlinear projection of high-dimensional data on a low-dimensional, regular topology but have also been used successfully for data partitioning [62, 63, 64]. In the context of SOMs, the topology of the target domain (and in this way the structure of the neural network) needs to be defined (commonly a $n \times m$ regular grid). Partitioning algorithms may produce hard clusters (e.g., k-means) or fuzzy cluster (e.g., fuzzy c-means). As an example, the popular k-means clustering algorithm is described in the following in more detail.

The k-means, k-medians, and k-medoids clustering algorithms define for each of the k clusters a reference center. The different approaches differ only in the way the cluster centers are defined. k-means clustering uses the mean data vector — i.e., the average value in each data dimension over all data items of a cluster — as cluster center. Analogous, the k-median algorithm employs the median data vector as cluster center. The k-medoid clustering algorithm selects the data item with the smallest sum of distances to the other items in a cluster as cluster center and is particularly useful in cases where a meaningful mean or median cannot be defined. During clustering, data items are assigned to clusters such that the intra-cluster variance (i.e., the mean-squared error) is minimized, i.e., the algorithm minimizes the sum of distances over the data items to the corresponding cluster centers. In practice, k-means, k-median, and k-medoid clustering are commonly implemented via an expectation-maximization (EM) algorithm. The basic structure of the EM algorithm is as follows:

1) Create k clusters and assign data items randomly to them.
2) Recalculate the centers of the k clusters.
3) Determine for each data item the closest cluster center.
4) Reassign data items to the according clusters.
5) If any data item has been reassigned to a new cluster center then go to step 2, otherwise stop. An alternative convergence criteria could be defined as a minimum decrease in the mean squared error.

The EM algorithm is iterative and stochastic, i.e., the algorithm may produce different clusters each time it is executed depending on the initial random assignment of data items to clusters. In practice, the algorithm is, therefore, often repeated several times to increase the likelyhood that a close to optimal solution is found. The described EM algorithm is in practice relatively stable assuming that the number of repeats is large enough.

Hierarchical clustering algorithms produce a nested series of partitions. Hierarchical clustering results are commonly described via a so-called dendogram, i.e., a hierarchical tree. The root of a complete dendogram represents the complete data and each leaf represents one data item. Each inner node of the dendogram, hence, defines a cluster of a number of data items. In practice, binary tree representations — i.e., trees in which each regular node has exactly two children — are most commonly used. The clustering hierarchy may be computed using agglomerative or divisive approaches. Agglomerative clustering algorithms use a bottom-up approach to compute a hierarchy of the data via subsequent merging of data items and clusters to define larger clusters. Divisive clustering algorithms use a top-down approach, i.e., they compute the data-hierarchy via subsequent splitting of the data into smaller clusters until all data items have been separated. In the context of this thesis, only bottom-up clustering approaches are used. The basic algorithm for bottom-up clustering can be described as follows:

1) Calculate the distance matrix. The distance matrix contains all pairwise distances between all data items.
2) Place all data items to be clustered into the list **L** defining all items to be merged.
3) Join the two nodes or items that are closest to each other.
4) Remove the nodes/items that have been merged from **L** .
5) Add the newly created node to **L** .
6) If **L** contains more than one element then go back to 3, else stop.

The actual dendogram is created by retracing which items and nodes were merged by the algorithm. The described hierarchical clustering algorithm is iterative and deterministic, i.e., given the same input conditions the algorithm produces the same result each time it is executed (unlike the EM algorithm and SOMs described earlier). Depending on how the distances between clusters are defined in terms of their members, different hierarchical clustering algorithms can be defined, e.g.:

(i) *Single linkage clustering* defines the distance between two clusters as the shortest distance between two members of the two clusters.

(ii) *Complete linkage clustering* defines the distance between two clusters as the distance between the two furthest members of the two clusters.

(iii) *Average linkage clustering* defines the distance between two clusters as the average over all pairwise distances between the data items of the two clusters.

(iv) *Centroid linkage clustering* defines the distance between two clusters as the distance between the centroids of the two clusters. The centroid of a cluster is commonly defined as the mean over all elements in the cluster.

2.7.2.2 Cluster Validation

Validation of clustering results and evaluation of an optimal number of clusters k is an important problem in data clustering. Validation of clustering results is often performed via investigation of statistical properties of the produced clusters as well as manually by expert users. Computational cluster validation techniques provide quantitative measures for cluster quality and promise to enable more objective and accurate validation of clusters. A survey of computational cluster validation techniques in the context for gene expression data analysis is provided by Handl et al. [65]. Cluster evaluation functions are commonly subdivided in external and internal measures.

External evaluation measures compare the result of a single clustering with a known set of unique as well as complete class labels (the so-called "*gold standard*" or "*ground truth*"). Based on the gold standard one commonly evaluates the purity and completeness of a classification result. The purity of a cluster is characterized by the fraction of data samples that were correctly assigned to the cluster, i.e., the fraction of the cluster taken up by the predominant class label. Completeness of a cluster can be described via the fraction of data samples belonging to the predominant class represented by the cluster that were actually assigned to the cluster. Popular external cluster quality measures include the F-measure [66] and the Rand index [67]. The F-measure is based on the idea of precision and recall commonly used in information retrieval, whereas the Rand index is based on counting the number of pair-wise co-assignments of data items. In case of the applications described in this thesis a gold standard is not known. Consequently, external cluster evaluation functions cannot be considered in this research.

Internal evaluation measures do not rely on a gold standard but evaluate a clustering based on the clustering result itself and the classified dataset. The most common internal cluster evaluation measures consider the compactness, connectedness, and/or separation of a clustering. Common measures for evaluation of the compactness of a cluster include the intra-cluster variance (i.e., the criterion optimized by the k-means algorithm) as well as average or maximum pairwise intra-cluster distances. The connectivity of a cluster can be described, e.g., via measures that compute for each data point of a cluster to which degree its n nearest neighbors have been placed into the same cluster [65]. Measures describing the separation of clusters frequently define average weighted inter-cluster distances, where the distance between two cluster is commonly defined as the distance between cluster centroids or as the minimum distance between data items belonging to a cluster (the single link criterion). Other common internal cluster quality measures, such as the Dunn Index [68] or the Silhouette Width [69], combine cluster compactness and separation criteria. The cluster quality measures described so far are general measures in the sense they do not employ any specific knowledge about the underlying data.

In the literature, several cluster quality measures specifically designed for the cluster analysis of gene expression data have been proposed. The *Figure Of Merit* (*FOM*) is an internal measure for gene clustering proposed by Yeung et al. [70], extended by Datta and Datta [71], that employs explicitly the redundancies and correlations often present in gene expression data. In the case of clustering of cells in 3D gene expression data, the level of redundant information is generally low. As a result, FOM and analysis techniques such as the overabundance analysis proposed by Ben-Dor et al. [72] are often not appropriate for the gene expression clustering applications described in this thesis but may be interesting when the cells of the embryo are to be classified based on the information of a very large number of genes. To the best of the authors knowledge, none of these existing cluster quality measures directly employ the fact that genes are expressed in characteristic spatial patterns.

Internal cluster quality measures have also been used to estimate the number of clusters k in a dataset. Estimation of an optimal k is usually done by computing a series of clustering results for an increasing number of clusters k. If a clustering algorithm and internal evaluation measure are adequate for the data to be classified, an optimal value of k can often be identified as a knee (or elbow) of the resulting performance curve. Tibshirani et al. [73] introduced the *gap statistic*, a statistical procedure that formalizes this heuristic. Milligan and Cooper performed a Monte Carlo evaluation of thirty procedures for determining the number of clusters in a data set [74]. Existing cluster evaluation measures are designed to find a single "perfect" k. As shown later in Section 4.4 and 4.4.2, when clustering cells in a 3D gene expression dataset, one typically finds a series of valid values for k, rather than the one "perfect" k.

2.7.2.3 Applications of Data Clustering to Gene Expression Data

An overview of various methods for cluster analysis for gene expression data is provided by Jiang et al. [75]. Gene expression data clustering can roughly be subdivided into three applications:

(i) clustering of genes to identify co-expressed genes indicating genes of similar function [76];

(ii) clustering of data samples to identify different phenotype structures of cells, e.g., Alizadeh et al. employed clustering for identification of different tumor cell types [77]; and

(iii) biclustering, i.e., the clustering of genes and data samples at the same time to find subgroups of genes and data samples where highly similar activities are seen for the genes in the subset of data samples [78, 79].

Clustering results are most commonly visualized using scatter-plots, plots of statistics, and color table views with columns and/or rows sorted with respect to the clustering. The broad applicability of clustering to gene expression has led to the development of several commercial and publicly available tools for clustering and visualization of gene expression data [80, 81, 82, 83, 84, 85]. However, these tools are limited to what essentially are one-dimensional analyses of gene expression in homogenized populations of cells. They do not take account of spatial position nor the complex relationship of expression across neighboring cells, and are consequently not suitable for interactive visualization and exploration of 3D gene expression data as described in this thesis (see Chapter 4).

2.7.3 Ensemble Methods for Data Classification

Methods that integrate multiple learned models into a single classification of the data are often referred to as ensemble methods. In the context of data clustering, ensembles are commonly used to improve the quality and robustness of traditional analysis methods.

According to Diettrich [86], combining multiple classifications improves the classification because ensembles partly overcome three main problems of learning algorithms that output only a single hypothesis, i.e, (i) the statistical problem, (ii) the computational problem, and (iii) the representation problem. The statistical problem is a problem of insufficient training data resulting in different hypotheses with similar accuracy and, hence, high variance of the algorithm. A vote of all potentially good hypotheses increases the probability of choosing the best one. The computational problem refers to the algorithmic problem that due the use of heuristics, a classification algorithms may not be able to guarantee to find the best hypothesis, e.g., neural networks or decision tree algorithms. Neural networks are also none-deterministic, i.e., given the same input conditions the algorithm may produce different outputs each time it is executed. Algorithms that suffer from the computational problem are sometimes also referred to as algorithms with high computational variance. The representation problem describes the issue that the search space of the algorithm may not contain any hypothesis that are good approximations to the real problem. By defining a weighted combination of multiple hypothesis, a more accurate approximation of the real problem may be possible. For example, a k-means clustering in combination with Euclidean distance is known to produce only hyper-spherical clusters. As Fred and Jain demonstrated, a k-means clustering ensemble can detect clusters of arbitrary shape [87].

Ensemble methods have a wide range of applications and are used in practice, e.g., for: (i) knowledge reuse, i.e., several classifications are available which a user seeks to integrate into a common classification; (ii) distributed data mining, e.g., in security sensitive applications where original data cannot be shared between different participating parties, or to (iii) improve quality and robustness of classifications [88]. In the field of pattern recognition, ensembles of different classifiers are used, e.g., to improve the quality and reliability of hand writing recognition [89]. In data clustering research, cluster ensembles help to improve the quality and robustness of traditional analysis methods, such as k-means [87], as well as to improve the robustness and stability of instable classifiers, such as neural networks [90].

In this thesis, the basic principle of ensembles is used to enable accurate classification of particle beams and high-performance analysis of extremely large 3D particle datasets. The beam path analysis algorithm described in Section 5.6 first analyzes each timestep separately to detect bunches of interest at each timestep and then merges the information of this ensemble of particle bunch classifications to define a consolidated description of the data. Using this approach leads to (i) increased performance of the analysis, (ii) improvement of the quality of initial analysis results, and allows for (iii) effective data reduction enabling efficient analysis of the complete timeseries.

Chapter 3

Background: 3D Gene Expression and Laser Wakefield Accelerator Simulation Data

This chapter introduces the two main application areas discussed in this thesis, i.e., (i) analysis of 3D gene expression data and (ii) analysis of laser wakefield particle accelerator simulations. Section 3.1 discusses the relevant background in developmental biology and describes the 3D gene expression data acquisition pipeline. Section 3.2 discusses the concept and simulation of laser wakefield particle accelerators and concludes with an overview of related work in data analysis in accelerator physics. The proposed novel frameworks for knowledge discovery from 3D gene expression data and laser wakefield particle accelerator simulations are discussed in Chapter 4 and 5 respectively

3.1 Introduction to 3D Gene Expression

Animal embryos comprise dynamic 3D arrays of cells that express gene products in intricate spatial and temporal patterns that determine the shape and form of the developing animal. The goal of the Berkeley Drosophila Transcription Network Project (BDTNP) [91] is to decipher how the patterns of gene expression underlying animal development are directed by the regulatory information contained in DNA sequences. To achieve this goal, the BDTNP has chosen the fruitfly *Drosophila melanogaster* as model organism. To date, studies of animal gene expression have not captured 3D context with cellular resolution. The BDTNP has developed methods to extract 3D PointCloud datasets from imaging data. PointCloud datasets provide for the first time information about the expression of genes with cellular resolution in three-dimensional space for complete fruitfly embryos. These novel datasets promise to enable the systematic characterization of the complex regulatory interactions and expression patterns already present prior to gastrulation.

Section 3.1.1 introduces genetic regulatory networks and their role in the control of embryo development. Section 3.1.2 describes the early development of *Drosophila* embryos. The regulation of the embryonic development of *Drosophila melanogaster* is described in Section 3.1.3. The development of *Drosophila* embryos and the required genetic fundamentals are highly complex and in the following only a brief overview can be provided. For more detailed information about the development of *Drosophila Melanogaster* see [92, 93]. Having introduced the essential background in developmental biology, Section 3.1.4 afterwards describes the BDTNP's pipeline for image acquisition and generation of 3D PointCloud data.

3.1.1 Genetic Regulatory Networks and Embryo Development

All cells of living organisms contain *deoxyribonucleic acid* (*DNA*), which encodes the genetic information of the organism. *Genes* are functional subsequences of the DNA. Most genes code for the amino acid sequences of proteins and additional *cis*-regulatory elements that help to determine in which cells the gene's product will be expressed.

Biosynthesis of proteins based on the information stored in genes is a fairly complex process that can be divided into two main subprocess, i.e., *transcription* and *translation*. In the transcription process the genetic information of a gene is copied by *RNA-polymerase* into a strand of *ribonucleic acid (RNA)*, the so called *messenger RNA (mRNA)*. The main function of mRNA is to transfer the genetic information of a gene from a chromosome to the *ribosomes* where the translation process is performed. During translation, nucleic acids are paired by *transfer RNA (tRNA)* molecules with amino acids. Finally a new strand of amino acids with a peptide backbone is build up which coils afterwards into a three-dimensional structure of minimal energy, completing the creation of a new protein molecule. The term *gene expression*, hence, refers to the amount of protein produced within a cell based on the genetic information of a gene.

An important class of protein coding genes are developmental regulatory *transcription factors* that function by binding to *cis-regulatory sequences* in many genes and direct their patterns of gene expression. Via binding or reverse binding, these transcription factors can promote or inhibit the expression of genes with the transcription factors themselves being proteins coded by genes. Complex *genetic regulatory networks* are built up where cascades of differently expressed transcription factors ultimately regulate all genes' expression. These networks guide the devel-

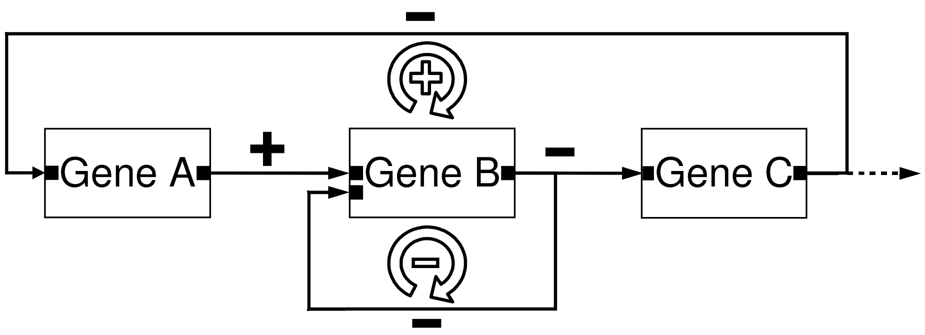


Figure 3.1: Example genetic regulatory network.

opment of all living organisms with the characteristic spatial and temporal patterns of regulatory transcription factors defining the body plan of the developing animal.

Simplified, genetic regulatory networks can be described by directed graphs in which genes are interacting via messages (i.e., proteins) transferred between them. These protein messages can either increase (+) or suppress (-) gene expression. The three main elements of genetic regulatory networks are *linear interaction*, *positive feedback loops*, and *negative feedback loops*. Linear interactions simply define linear chains of gene interactions with no reverse effects. Feedback loops define more interesting elements of regulatory networks in which a gene is directly or indirectly regulated by itself. Depending on whether the activity of a gene leads to self-activation or self-inhibition, one in general distinguishes between *positive feedback loops* and *negative feedback loops*, respectively. In practice, negative and positive feedback loops can be distinguished simply via the number of negative interactions they contain. Positive feedback loops contain an even and negative feedback loops an uneven number of negative interactions. Positive and negative feedback-loops differ not only in their structure but more importantly in their behavior. Negative feedback-loops tend to produce a balanced situation after some time. Positive feedback-loops on the other hand can have multi-stationary states.

Figure 3.1 shows a simple example illustrating the main elements of genetic regulatory networks. A negative feedback-loop is constituted for gene B via direct product inhibition. For gene A, a slightly more complex positive feedback-loop can be seen. Here gene A indirectly inhibits its own inhibitor gene C by activating gene B which in turn inhibits gene C. As also illustrated in Figure 3.1, one gene is in general affected by several different transcription factors, resulting in highly complex, extremely dynamic systems of interacting feedback loops. The dynamics of genetic regulatory networks are also affected by various other factors, such as the locations of the genes on the chromosomes, physical and chemical properties of transcription factors, as well as external factors like temperature and concentration of various chemical compound. The complex structure and dynamics of genetic regulatory networks make analysis and in particular prediction of the behavior of regulatory networks under varying conditions a challenging task.

3.1.2 Early Embryonic Development of *Drosophila melanogaster*

The embryonic development of *Drosophila melanogaster* is subdivided into seventeen stages. The different stages are defined based on important, easily distinguishable features of the embryo formed during the development process. The research presented in this thesis focuses on the analysis of *Drosophila* embryos at stage five of embryo development.

At the beginning, the egg itself includes mitochondria, ribosomes, storage material, proteinaceous yolk, and vesicles of lipid. In the *Drosophila* egg, nuclei divide and migrate in a common cytoplasm instead of complete cleavage of the cells. During the first two hours, the embryo is not defined by individual cells but forms a multinucleated mass of cytoplasm, also called a *syncytium*. At this stage, proteins can spread freely within the egg, a fact that is utilized in early pattern formation during which gradients of protein concentration provide positional information within the embryo (see Section 3.1.3).

At the beginning of embryo development, the nuclei are roughly located in the center of the embryo. During stage one and two of embryo development, the first seven cycles of nuclear division take place. Zygotic nuclei divisions proceed synchronously within about every nine minutes during this phase. After the fifth division, the nuclei start to move peripherally. With the end of the eighth syncytial division, the majority of the zygotic nuclei are evenly arranged on the outer surface of the embryo. Some nuclei drop out or stay at the center of the embryo, the so-called *yolk nuclei*.

During stage three of embryo development, the ninth syncytial devision is performed. In this stage, polar buds are formed around the nuclei located at the posterior pole of the embryo.

During stage four of embryo development — the *syncytial blastoderm stage* — the tenth to thirteenth syncytial divisions of the blastoderm nuclei are performed. The second division of the polar buds takes place during the tenth syncytial division. The buds pinch off afterwards and form twelve to fourteen *pole cells*. The pole cells and blastoderm nuclei behave from then on differently. The pole cells divide about twice as much and begin with protein synthesis early. This behavior illustrates the first differentiation in the embryo. Stage four of embryo development is completed with the onset of cellularization.

During stage five of embryo development — the *blastoderm stage*— the nuclei that have migrated to the outer periphery of the egg form a monolayer of cells, the *blastoderm*. Introgression of membrane furrows separate single blastoderm nuclei to form separate cells, which all together build up the blastoderm. Blastoderm formation is a complex process and depending on the progress of the formation of the ingrowing membrane, developmental stage five is usually subdivided into several substages. While the plasma membrane extends centrically into the egg, nuclei are not directly separated completely but for some time there are still channels between the ascent cells and the yolk. These channels are large enough so that even large molecules, like proteins, can pass through them. Interaction between the blastoderm nuclei is in this way still directly possible. Molecules pass from one nucleus to others and act as a kind of signal transmitters.

During the prolonged interphase between the beginning of blastoderm formation and complete separation of the blastoderm cells, the body plan of the embryo is laid out in the blastoderm cells, i.e., each cell of the blastoderm is assigned a defined role in the further development of the fly. The function of a cell is thereby determined by which genes are expressed at which levels in the cell. For the research presented in this thesis only the cells in the blastoderm are of interest, a fact that is utilized in the visualization to create effective two- and three-dimensional model representations of the embryo (see Section 4.2.1).

During developmental stage six and seven, a subset of ventral cells of the embryonic blastula fold in to create a two-layered embryo. This process is also referred to as *gastrulation*. The inner cell-layer forms during later development stages inner structures, like organs, in the embryo. Step by step, the different parts of the new fruitfly are created in the following development stages.

3.1.3 Regulation of Embryo Development

The body-plan of a fly is defined through complex genetic regulatory networks that result in the formation of complex spatial and temporal patterns of gene expression. The function of a cell is determined based on which genes are expressed in it at which levels. Together, the patterns of a large number genes ultimately control the fate of all cells. To be able to form the different parts of a fly's body in the proper places, some form of dynamic positional information that acts via direct interaction with the genetic information of both, the egg and the sperm, is needed.

The mother sets up several systems of *morphogen gradients* in the egg, defining the main axes of the egg. *Morphogens* are diffusible proteins located in special parts of the embryo where the concentration of a morphogen is high at its source and decreases with increasing distance from the source. The different morphogens act as transcription factors or translational repressors (i.e., a protein that inhibits translation of a gene), regulating the expression of other genes. A morphogen can have one main or several different target genes. Via complex genetic regulatory networks, the expression of many different genes are affected either directly or indirectly by mor-

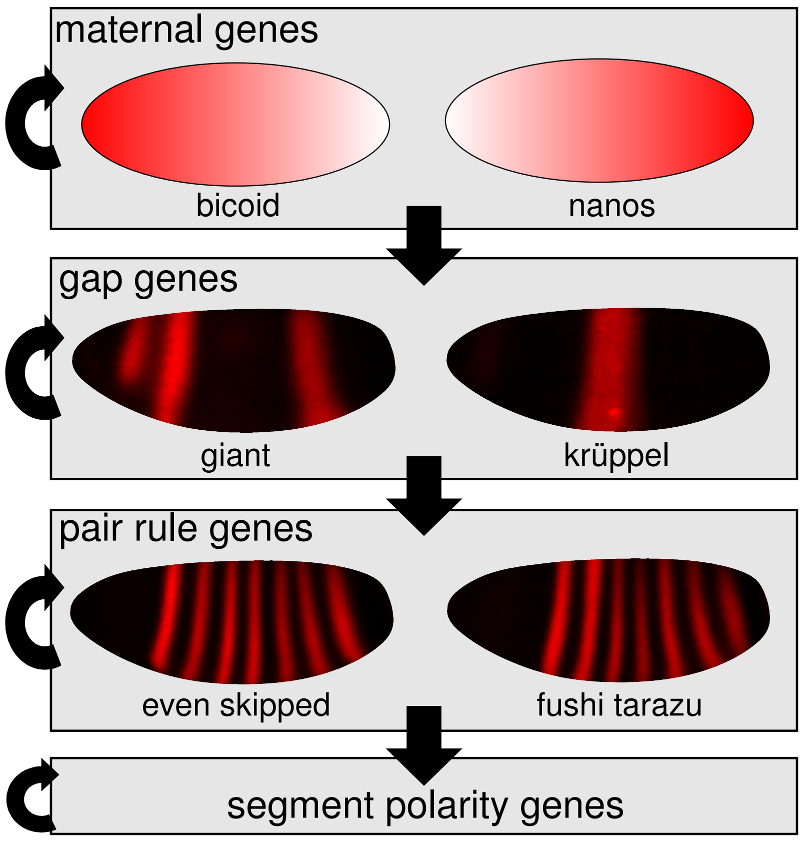


Figure 3.2: Illustration of the hierarchy of segmentation genes.

phogens. Depending on the local concentration of morphogens, the elicited expression response of genes may vary, resulting in the formation of complex spatial patterns of gene expression. Via this complex mechanism, gradients of morphogen concentration function as positional information in the embryo. To control embryo-development, the *Drosophila* mother sets up four morphogen gradient systems in the egg:

(i) an *anteroposterior gradient* of the *bicoid* protein,

(ii) a *posterior system* with *nanos* as the effective agent,

(iii) a *terminal system* that functions as the third determination of the anterior-posterior axis and which induces the terminal parts of the embryo,

(iv) a *dorsoventral system* that determines the dorso-ventral axis.

As a direct response of zygotic genes to these morphogen gradients the, embryo gets subdivided into segmental entities. During the first three hours of embryo development, the embryo is subsequentially subdivided into smaller domains. Segmentation genes are responsible for the correct development of the corresponding regions of the embryo and are expressed in complex spatial patterns. Genetic analysis have revealed a hierarchy of segmentation genes consisting of four main steps (see Figure 3.2). *Maternal genes*, like bicoid and nanos, define the first level of this gene-hierarchy and make up the described morphogen gradients. Afterwards, *gab genes*, *pair rule genes*, and *segment polarity genes* are expressed. The expression of each class of genes and, hence, the correct localization of the gene expression patterns is regulated by genes of the previous class of genes as well as through cross regulation between members of the same gene class.

Gab genes subdivide the embryo into broad domains. The pair rule genes further refine the subdivision of the embryo based on the expression patterns of the gab genes. Pair rule genes are expressed in patterns consisting of seven stripes. One can further distinguish between primary pair rule genes —i.e, genes that translate the spatial information of maternal and gab genes into the characteristic stripe patterns— and secondary pair rule genes, i.e., genes with a seven-stripe pattern regulated by primary pair rule genes. Pair rule genes are only transiently expressed and are responsible for activation of segment polarity genes. The segmentational information is in this way transmitted form the pair rule genes to the segment polarity genes. At the onset of gastrulation, segment polarity genes start to express in fourteen stripes. Segment polarity genes stabilize and refine the striped pattern and are responsible for definition of the fourteen *parasegments* of a *Drosophila* embryo. Gab and pair rule gene expressions fade away at the time at which segment polarity genes adopt their functionality.

3.1.4 From *Drosophila* Embryos to 3D PointClouds

To provide a quantitative description of the complex patterns of gene expression in the early *Drosophila* embryo, the BDTNP has developed a data processing pipeline for extracting precise measurements of spatial gene expression patterns in three-dimensional space (see Figure 3.3). *Drosophila* embryos are first fluorescently stained and imaged using two-photon microscopy (see Section 3.1.4.1). Each image is segmented to extract information, such as nuclear positions and volumes as well as expression values in the neighborhood of each nucleus for the chosen genes [94] (see Section 3.1.4.2). The resulting *Single PointCloud* files contain information about either protein or mRNA expression of the genes. A detailed description of the image acquisition and segmentation process was provided by Luengo Hendriks et al. [94].

It is not practical to obtain the expression of more than a few genes in a single embryo, due to the limited number of different distinguishable fluorophores as well as the difficulty in adding multiple labels to embryos. To allow relationships between multiple transcription factors and their target genes to be compared in a common coordinate framework, Single PointClouds are

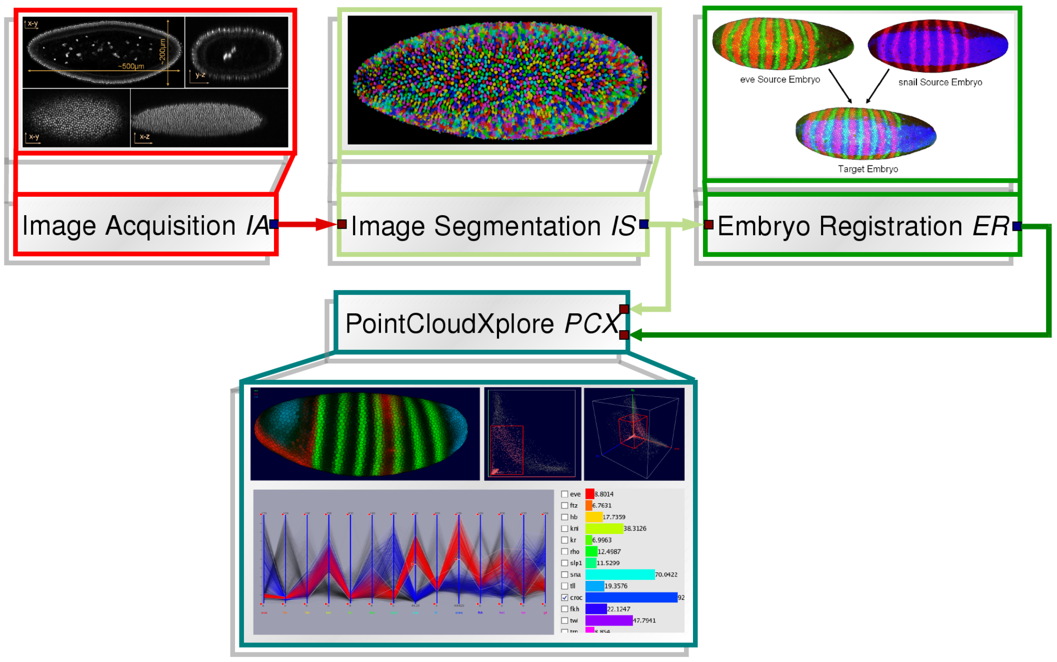


Figure 3.3: Data acquisition, processing, and visualization pipeline for 3D gene expression data. Three-dimensional images, each containing a whole embryo, are transformed into Single PointCloud files containing information about cell positions and the expression of the measured genes. Multiple Single PointCloud files are registered to compute Virtual PointCloud files with information about the expression of a larger number of genes. PointCloudXplore (PCX) is used for visual analysis of Single as well as Virtual PointClouds. Arrow color indicates the type of data transferred between the different parts of the pipeline with red being raw image data, light green being Single PointClouds, and dark green being Virtual PointClouds.

registered to create *Virtual PointClouds* (also called *Expression Atlases*) describing the average expression of many genes in a single virtual embryo model [161] (see Section 3.1.4.3). The embryo registration process is described in detail in [161]

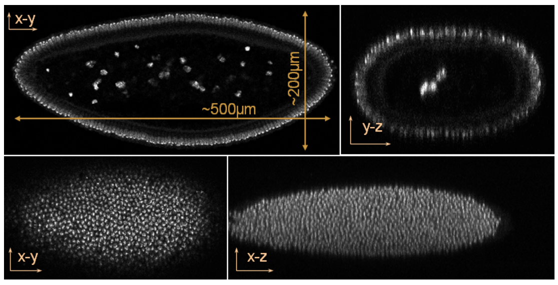
The visualization system PointCloudXplore (PCX) is used for visual analysis of Single as well as Virtual PointClouds [157, 159] (see Section 4.2 for details). The following parts of this section describe the different steps of the introduced processing pipeline, consisting of image acquisition (see Section 3.1.4.1), image segmentation (see Section 3.1.4.2), and embryo registration (see Section 3.1.4.3).

3.1.4.1 Image Acquisition

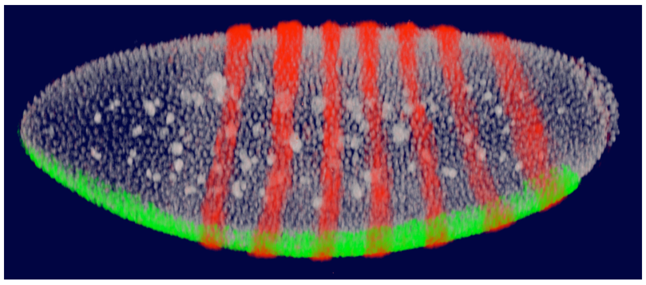
The image acquisition process can be divided into two main steps: (i) embryo preparation, and (ii) imaging. The preparation of embryos for imaging consists of two main steps, called *staining* and *mounting*. In the staining process, appropriate fluorescent color markers are introduced in the embryo in order to allow measurement of DNA, mRNA, and/or protein concentrations. Markers are detectable, specific probes that attach with high selectivity to specific molecules in cells of the embryo. In the mounting process, the embryo is fixated for the imaging process to avoid imaging errors due to movement of the embryo and to ease focusing of the microscope on the embryo.

After preparation, embryos are imaged using a Zeiss LSM 510 laser-scanning microscope. For each embryo, a stack of 120 to 140 images is created. Each image contains one slice of the embryo at a fixed z-axis position. Each image slice has a resolution of 1024×1204 pixels with a pixel size of $0.45 \times 0.45 \times 1.5\mu m$. Fluorescence intensities for the nuclear stain and labeled gene products are separate values captured on a per pixel basis. A single image stack contains one whole embryo and has a size of ≈ 0.5GB.

Figure 3.4 shows a set of example raw data images of a *Drosophila* embryo. *Drosophila melano-gaster* embryos at stage five of embryo development are approximately $500\mu m$ long and $200\mu m$ wide and contain a monolayer of $\approx$5000 - 6000 tightly packed nuclei.



(a)



(b)

Figure 3.4: a) Slices of raw *Drosophila* microscopy image data. b) Volume rendering of *Drosophila* 3D microscopy image data with nuclei (white) and two genes (red and green)

3.1.4.2 Image Segmentation

The imaging data provides information about the expression of genes on a per-pixel basis. In the second step of the BDNTP's processing pipeline, 3D images — each containing a singel embryo — are segmented to extract cell positions, cell and nuclear volumes, surface normals, and gene expression data on a per-cell basis.

Three-dimensional imaging, in particular for high-throughput applications, is a complex task resulting in several imaging problems that make image segmentation a challenging but now well-solved task. Besides the tight packing of nuclei, the anisotropic resolution of the microscope is

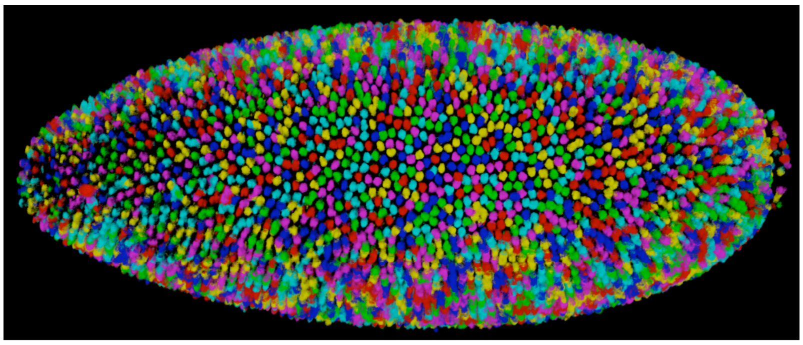


Figure 3.5: Visualization of segmentation results with segmented nuclei colored randomly. (*Image courtesy of Gunther H. Weber [97]*)

problematic. Image resolution along the optical z-axis is about three times lower than in the optical xy-plane. Penetration loss, yolk DNA, the pole cells, and the fact that DNA density is not uniform across nuclei introduce additional complications.

Novel image segmentation and analysis algorithms were developed specifically tailored to detect the monolayer of nuclei forming the *Drosophila* blastoderm [94]. Local maxima of the smoothed *Sytox* image showing the total DNA are used as cell markers. Many nuclei contain two markers instead of just one. A pruning algorithm is used to remove false cell candidates. Using a watershed-based segmentation approach, the markers are extended to fill the nuclei and the nuclei are extended to fill the cells. Together with the image segmentation, the expression of the stained genes is quantified. Figure 3.5 shows a volume rendering of an example segmentation mask with individual nuclei colored randomly. The segmentation process is described in detail in [94].

The segmentation produces so-called *Singel PointCloud* files containing information about the position, volume, surface normal, and expression values for each cell of a single embryo. A Single PointCloud file (stored in ASCII format) requires only about 1MB of storage space compared to ≈500MB for the raw image data (stored in binary format). The specification of the file format for Single PointClouds is available from [95] (see also [96] for information about the Virtual PointCloud file format).

3.1.4.3 Embryo Registration

Due to the limited number of different distinguishable fluorophores as well as the difficulty in adding multiple labels to embryos, it is not practical to obtain the expression of more than a few genes in a single embryo during imaging. To allow relationships between multiple transcription factors and their target genes to be compared in a common coordinate framework, Single Point-

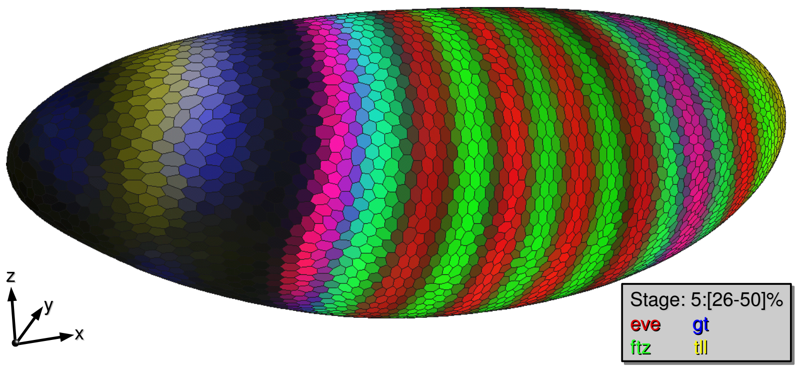


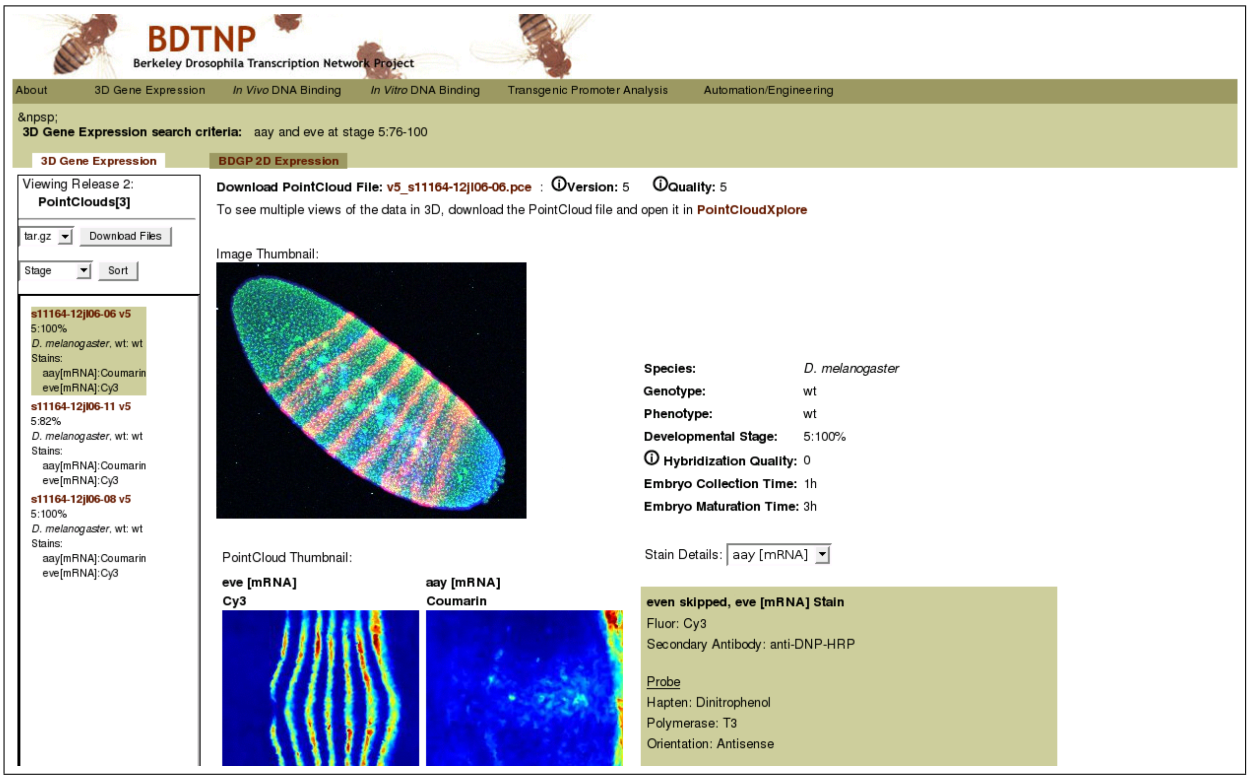
Figure 3.6: Visualization of a three-dimensional virtual embryo model showing the expression of four selected genes via color.

Clouds are registered into a *Virtual Embryo* using both morphology and a common reference gene to determine cell correspondences [98][161].

Because the spatial expression patterns of genes change rapidly during stage five of embryo development, the BDTNP defines the developmental stage of embryos based on invagination of cell membranes and groups PointClouds into six temporal cohorts [94]. For temporal comparisons, different cohorts are matched using cellular flow fields that predict the positions of individual cells at each time point [99][161]. This method enables BDTNP researchers to follow gene expression levels within a particular cell over time using only data measured in fixed embryos. Hence, each cell in the Virtual Embryo contains gene expression levels for each of the six time steps. This cellular-level link between embryos of different ages makes it possible to study the development of gene expression patterns over time, as well as to use an mRNA expression pattern as an approximate substitute for a later protein expression pattern, when suitable protein data is not yet available [161].

Embryo registration also greatly reduces the number of images needed to compare the expression of many genes. A comparion of the expression patterns of N genes based only on Single PointClouds would require at least one hybridizations for each gene pair, i.e., $O(N^2)$ hybridizations. To create large scale models of thousands of genes, several million hybridizations would be necessary. Registration of embryos makes it possible to construct such a complete picture with only $O(N)$ hybridizations.

The visualization system PointCloudXplore *PCX* is used for visualization of both, Single and Virtual PointCloud datasets. Figure 3.6 shows an example visualization of a virtual embryo model. The expression values of the genes *even skipped (eve)*, *fushi tarazu (ftz)*, *giant (gt)*, and *tailless (tll)* are shown via color. Similar to the staining during imaging, each gene is assigned a



&npsp;
3D Gene Expression search criteria: aay and eve at stage 5:76-100

Viewing Release 2:
PointClouds[3]

s11164-12jl06-06 v5
5:100%
D. melanogaster, wt: wt
Stains:
aay[mRNA]:Coumarin
eve[mRNA]:Cy3

s11164-12jl06-11 v5
5:82%
D. melanogaster, wt: wt
Stains:
aay[mRNA]:Coumarin
eve[mRNA]:Cy3

s11164-12jl06-08 v5
5:100%
D. melanogaster, wt: wt
Stains:
aay[mRNA]:Coumarin
eve[mRNA]:Cy3

Download PointCloud File: v5_s11164-12jl06-06.pce : **Version:** 5 **Quality:** 5
To see multiple views of the data in 3D, download the PointCloud file and open it in **PointCloudXplore**

Species:	*D. melanogaster*
Genotype:	wt
Phenotype:	wt
Developmental Stage:	5:100%
Hybridization Quality:	0
Embryo Collection Time:	1h
Embryo Maturation Time:	3h

even skipped, eve [mRNA] Stain
Fluor: Cy3
Secondary Antibody: anti-DNP-HRP

Probe
Hapten: Dinitrophenol
Polymerase: T3
Orientation: Antisense

Figure 3.7: Screenshot of the online BDTNP 3D gene expression database.

basic color value with gene expression values shown via brightness of color. An introduction to PCX is provided in Section 4.2.

The BDTNP has made its PointCloud database[1] available to the public, enabling researchers to efficiently search and access 3D gene expression datasets, i.e., Single as well as Virtual PointClouds. Besides the PointCloud files themselves, researcher here have access to detailed information about how individual PointClouds were acquired and the information they contain. For Single PointClouds, information is provided, e.g., about the species, genotype, phenotype, and developmental stage of the imaged embryo, the stained genes and used markers, as well as thumbnail images with a preview of the gene expression patterns, nuclear density, and image orientation. For Virtual PointClouds, information about the PointClouds used in the registration process, correspondence files, and a table with previews of all available expression patterns is provided. Figure 3.7 shows a screenshot of the BDTNP's 3D gene expression database. Via their webpage the BDTNP has also made PCX available to the public free of charge.

[1]The BDTNP 3D gene expression database is available online at: `http://bdtnp.lbl.gov/Fly-Net/bioimaging.jsp`

3.2 Introduction to Laser Wakefield Particle Acceleration

This section describes the concept, simulation, and analysis of laser wakefield particle accelerators (LWFAs). LWFAs are a special type of particle accelerator in which a laser pulse is used to drive a wave in a plasma. Similar to a surfer who accelerates while riding a wave, particles can become trapped in the plasma wave and accelerated to high energy levels.

Particle accelerators use electric fields to accelerate electrically charged particles (e.g., electrons) to high speeds and contain them. A common *cathode ray tube* (CRT) television is an example of a simple type of accelerator.

In science today, particle accelerators are among the most powerful devices enabling scientific discovery. In fundamental science, particle accelerators are used to study the structure of matter and continue to shape the scientific models of the universe. In the context of material science and biology, terahertz and X-ray radiation produced via particle accelerators is used to, e.g., decipher molecular structures. Other applications of particle accelerators include medical imaging and radiotherapy. A recent report by the Office of High Energy Physics of the US Department of Energy (DOE) [100] outlined the applications, challenges, and science opportunities of high-energy physics in the next ten years. In its summary the report states: *"Accelerators and experiments at the Energy Frontier are expected to make major discoveries leading to an ultimate understanding of the theory of particles and their interactions. They will address key questions about the physical nature of the universe: the origin of particle masses, the existence of new symmetries of nature, extra dimensions of space, and the nature of dark matter."*. This statement illustrates the importance of particle accelerators in science discovery today.

Conventional accelerators use metallic cavities to shape radiofrequency electromagnetic waves to produce accelerating fields. The maximum accelerating field that can be achieved using this technology are limited in part due to electric breakdown of the metallic cavities. Therefore, with the need to achieve ever higher energy levels, the size of conventional accelerators increases. For example, using conventional accelerator technology a TeV-class electron-positron linear collider requires accelerators at the length of about 20km. The world largest accelerator is the large hadron collider (LHC) at CERN, the European Organization for Nuclear Research, near Geneva at the border between France and Switzerland. The precise circumference of the LHC accelerator measures 26,659 meter and is equipped with a total of 9300 magnets [101].

The following subsections provide an introduction to laser wakefield particle acceleration (see Section 3.2.1) as well as the simulation of laser wakefield accelerators (see Section 3.2.2) and the data produced by a simulation (see Section 3.2.3). Subsection 3.2.4 describes related work on data analysis in LWFA physics. The proposed framework for knowledge discovery from laser wakefield particle acceleration simulation data is described in Chapter 5.

3.2.1 Plasma-based Acceleration

Plasma-based particle acceleration is a concept that promises to alleviate the "curse of size" (and cost) of next-generation particle accelerators. Rather than metallic cavities and electro magnets, plasma-based accelerators utilize an electric field generated by an electron plasma wave to accelerate charged particles (e.g., electrons) to high energy levels. The concept of plasma-based accelerators was first proposed by Tajima and Dawson in 1979 [102] and has since evolved into an active field of research. LWFAs can easily achieve acceleration gradients of 10-100 GVm^{-1} [102, 103] allowing particles to be accelerated to high energy levels within short distances, e.g., 1 GeV particle beams were achieved using cm long plasmas [104, 105].

This work focuses on a specific type of plasma-based accelerator, the so-called laser wakefield particle accelerator (LWFA). The basic concept of an LWFA is to use a short ($\lesssim 100fs$), ultrahigh intensity ($\gtrsim 10^{18}W/cm^2$) laser pulse to drive waves in a plasma. When applied to a hydrogen plasma the radiation pressure of an intense laser pulse displaces the electrons while leaving the heavier ions stationary. Together with the space-charge restoring force of the ions, this displacement drives a wave (wake) in the plasma. If the wake is high enough in amplitude electrons can become trapped and accelerated by the plasma wave, and eventually decelerate again as they outrun the wake. The electric and magnetic fields that can be achieved in an LWFA are several thousand times stronger than in conventional accelerators [106] allowing particles to be accelerated to high energy levels within very short distance. Figure 3.8 shows a snapshot of an LWFA simulation illustrating the acceleration process.

In practice scientist distinguish between the LWFA and the self-modulated laser wakefield accelerator (SMLWFA). The main difference between an LWFA and an SMLWFA is the ratio between laser pulse length (L_{laser}) and the plasma wavelength (λ_p). In an SMLWFA L_{laser} is larger than λ_p causing the laser pulse to undergo a self-modulation instability whereas in an LWFA L_{laser} is usually smaller than λ_p. In case of the simulation data used in this thesis L_{laser} is usually in the range of $\approx [0.5 * \lambda_p, 1.5 * \lambda_p]$.

Alternative plasma-based accelerator concepts not considered in this research are the plasma wakefield accelerator (PWFA) and the laser beat-wave accelerator (LBWFA). A PWFA uses an electron beam rather than a laser pulse to drive the plasma wave(s). A LBWFA uses two long pulse laser beams with different frequencies to resonantly excite the plasma wave. As an extension to the LWFA and PWFA concept, multiple pulses (laser or electron pulses) have been used to enhance the wakefield amplitude. An overview of the different concepts for plasma-based

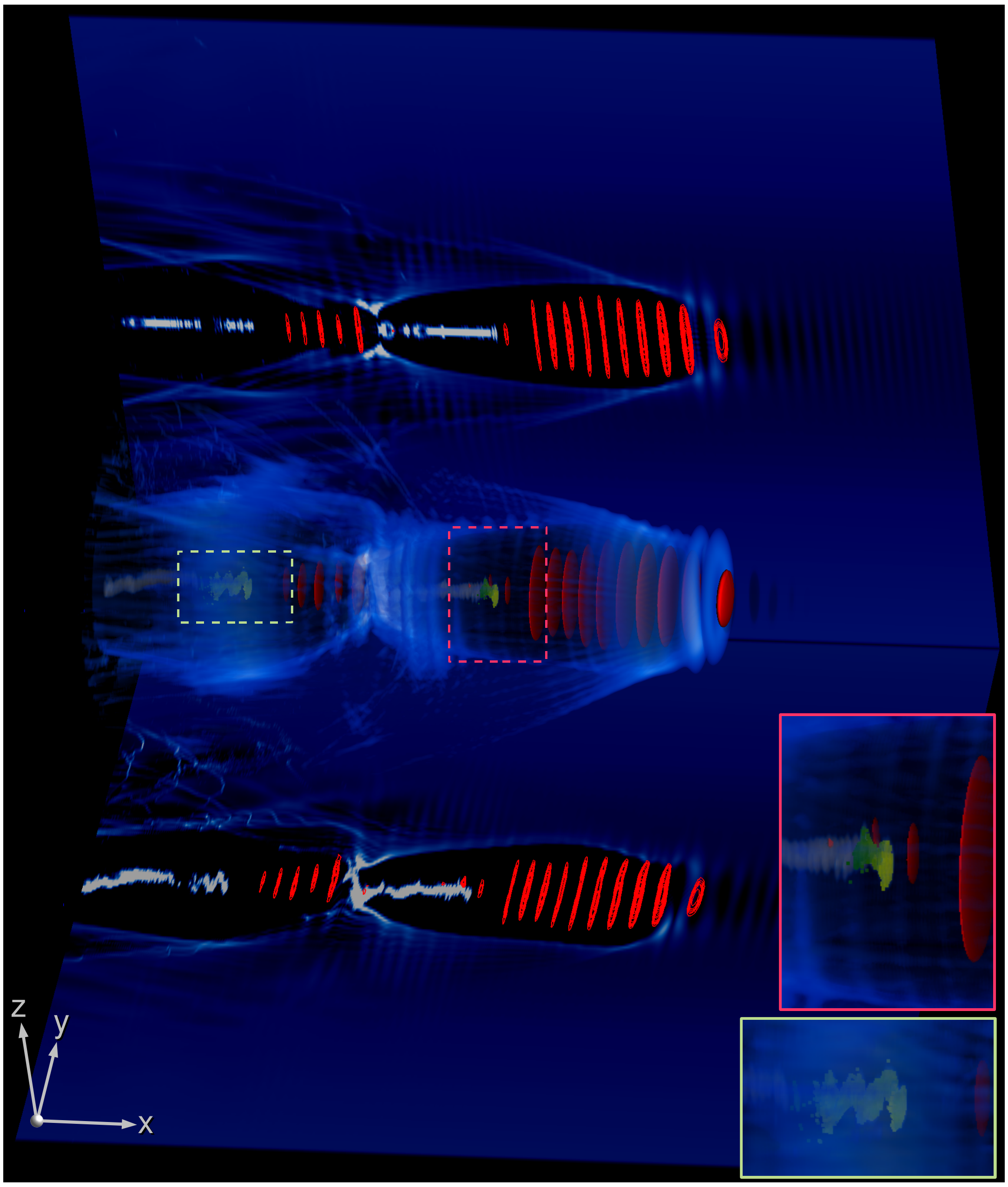


Figure 3.8: Volume rendering of the plasma density illustrating the three-dimensional structure of the wake (blue). A set of particles trapped in the wake that are accelerated to high energy levels ($px > 5 * 10^{10}$) are shown in addition in green/yellow with green being medium and yellow being high momentum in x direction px. Contours (red) of the electric field strength in z direction (E_z) illustrate the laser pulse (which moves along the x axis from left to right). The panel at the back and bottom show a slice through the center of the volume in x/y direction (at $z = 0$) and x/z direction (at $y = 0$) respectively. The inset views show close-ups of the two main bunches of accelerated particles.

acceleration is provided in a survey by Esarey et al. [103].

While LWFAs can easily achieve high acceleration gradients, the electron beams produced by LWFAs were for a long time of low quality, i.e., relatively low energy levels <200MeV with 100% energy spread [107]. Controlling the acceleration process is a challenging task due to the complex interactions between the laser pulse and the plasma and vice versa, i.e., the laser pulse and the plasma evolve together. It was not until recently in 2004 that three different groups were independently able to obtain percent-level energy spread 70-200MeV beams using intense laser pulses applied to millimeter-scale gas jets [108, 109, 110]. One method to achieve such high-quality bunches is through plasma channel guiding in which a preformed plasma channel — similar to an optical wire — guides the laser beam through the plasma [108]. Using this method, researchers at the LOASIS [111] program have demonstrated high-quality electron beams at 1GeV using cm long plasmas [104, 105]. Recently, Geddes et al. [112] described the use of plasma density gradients to control wake phase velocity and particle trapping in LWFAs demonstrating stable, low-momentum-spread bunches that would be suitable for use as an injector.

Analysis, understanding, and control of the complex physical processes of plasma-based particle acceleration is a challenging task and requires understanding of how particles become trapped and how the particle beams are formed and accelerated. These processes are best understood by tracing the particles that form a beam over time and investigate their temporal evolution [106, 113]. In real-world experiments it is, however, impossible to record the complete evolution of an experiment and much less to trace single particles within a plasma. Simulation of LWFA experiments is, hence, essential for the understanding of the acceleration process.

3.2.2 Simulation

To better understand nonlinear plasma response, beam trapping, self-consistent laser propagation, and beam acceleration — processes not accessible to analytic theory — LWFA experiments are computationally modeled. Simulation of LWFAs is essential for the understanding of the fundamental physics of plasma-based acceleration, understanding of processes and results observed in experiments, as well as for improvement of experiments. Simulation is also becoming increasingly important for planning of experiments and testing of new accelerator models. In the past, large-scale one-to-one simulations, benchmarked to experiments, enabled scientists to study and understand the physics of formation of narrow energy spread particle bunches from self-trapping at 0.1GeV and 1GeV [114]. Today, simulation is used for development of new accelerators involving, e.g., controlled injection to increase beam quality [115].

Traditionally, explicit particle-in-cell (PIC) simulations are used to model LWFA experiments [116]. PIC simulations — using, e.g., VORPAL [117], OSIRIS [118] and others simulation codes [119] — self-consistently include fields and interactions of the laser, plasma, and bunch. This is important in the case of plasma-based accelerators because the laser pulse is simultaneously shaped by its interactions with the plasma as it drives the wake, i.e., both, the laser and the plasma, evolve together while in this process a large portion of the laser energy is transferred into the plasma. Balancing this process is the basis for the formation of a stable accelerating structure.

This work concentrates on datasets produced by VORPAL [117][2], a parallel, object-oriented plasma simulation code, which can model the behavior of charged particles in their self-consistent electromagnetic field. Both, the particle motion as well as the field evolution, can be modeled with various approximations, ranging from fluids to fully kinetic particles, or from an explicit electromagnetic to an electrostatic treatment of the field equations. The code has been successfully used across a number of disciplines within DOE's Office of Science, including ultra-high gradient laser-plasma acceleration of electron beams, electron cooling of heavy ion beams, and implicit electromagnetic treatment of plasma edge phenomena in tokamak plasmas.

When modeling LWFAs, VORPAL is typically used as PIC code (see Figure 3.9). In this method, collections of real charged particles are modeled as computational macro-particles that can be located anywhere in the computational domain. Each macro-particle represents a set of real particles while the weight of each macro-particle indicates the number of particles it represents. The electromagnetic field is spatially discretized via a regular grid. Particles are moved under via Newton-Lorentz force obtained through interpolation from the fields. The current carried by the moving particles is then deposited onto the simulation-grid to solve Maxwell's equations for the fields. VORPAL's design enables simulations to be run in multiple spatial dimensions, ranging from 1D to 3D. The large computational cost of 3D simulations requires thorough studies in 2D prior to a full 3D run. Being able to run the same code base with the same input configuration enables to run much more efficiently. With increasing level of parallelism used in simulations, efficient I/O becomes a major challenge. VORPAL uses parallel HDF5 [120] for I/O operations utilizing SUN's LUSTRE parallel file system installed at NERSC[3].

In order to accurately model an LWFA the simulation needs to resolve the laser wavelength (λ_{laser}; usually in the order of μ-meters) over the length of the accelerator (L_{accel}; usually in the

[2]*VORPAL = Versatile Plasma Simulation Code*; VORPAL is a software for performing relativistic, hybrid plasma and beam simulation developed by the Tech-X corporation (see online at: `http://www.txcorp.com/`).VORPAL is used by researchers from the LOASIS [111] program to simulate laser wakefield particle accelerators.

[3]*NERSC = National Energy Research Scientific Computing Center*; NERSC is a scientific computing facility for the Office of Science in the U.S. Department of Energy.

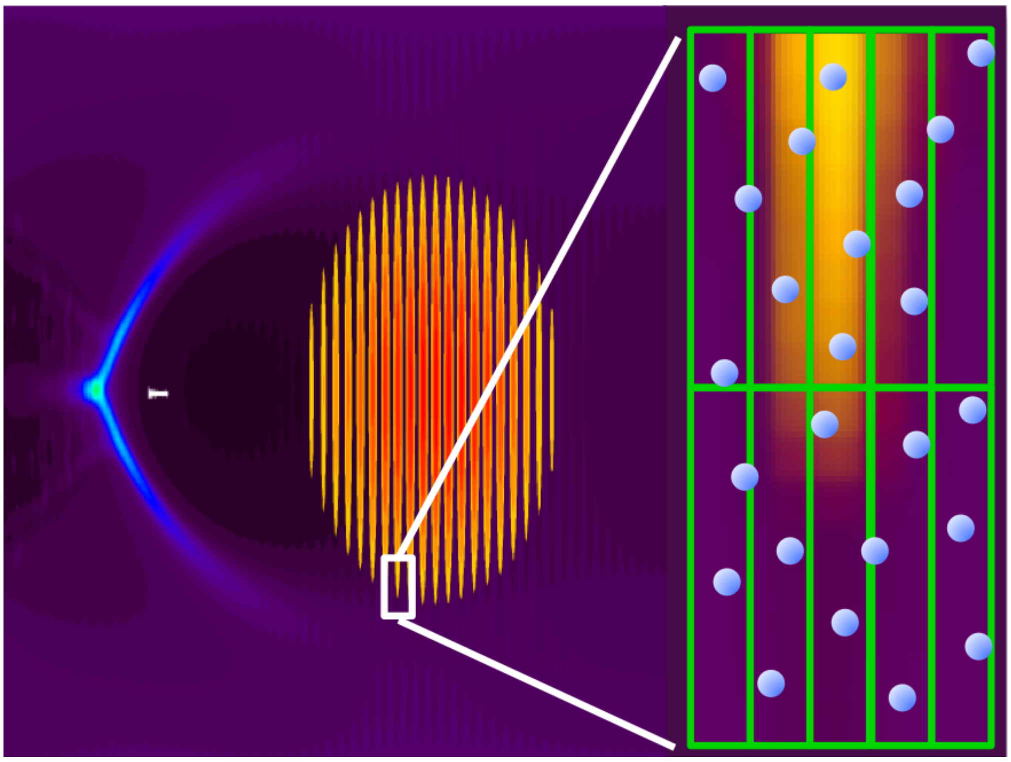


Figure 3.9: Particle-in-cell (PIC) simulation of a laser wakefield particle accelerator with the plasma density shown in lilac/blue and the laser shown in yellow/red. The inset plot shows a close-up view of the computational grid that resolves the laser period (yellow) and the plasma particles (blue). *(Image courtesy of Cameron G. R. Geddes)*

order of millimeters to centimeters and soon meters), i.e., the resolution of the simulation grid in longitudinal direction (x) must resolve λ_{laser}. In transverse direction (y and z) the grid-resolution must be high enough to resolve the structure of the plasma wave. Furthermore, each gird-cell should contain a sufficient amount of particles to avoid large numerical errors, i.e., the number of simulated particles should at least be equal to the number of cells of the simulation grid or larger.

In order to accurately model todays experiments via PIC simulations, researchers, therefore, use up to hundreds of millions of particles as well as grid-cells (e.g., $4000 * 300^2$ grid-cells per timestep in a moving window simulation, see details below) and a million timesteps. On current supercomputers, such a LWFA simulation requires hundreds of of processor hours using more than 10,000 processors. Even such high-resolution simulations are only an approximation. A real plasma contains hundreds of trillions of particles (rather than millions of macro-particles) and LWFAs produce X-ray radiation and other fields with wavelengths unresolved by current simulations. The requirements for simulation of meter-scale 10 GeV experiments are even stricter to ensure that laser pulse velocity at low density and beam quality for applications are resolved accurately. In combination with the increased length and width of the accelerator, these stricter requirements are expected to make simulation of next-generation LWFAs at least three to four orders of magnitude more costly than current experiments.

Due to the large computational resources required to accurately model a LWFA, it is not practical to simulate the entire cavity containing the hydrogen plasma at once. To save computational resources and storage space, VORPAL employs a moving window simulation approach. In this method, only a region around the laser pulse is simulated at each time step. As the laser pulse is traveling through the plasma the simulation window is moved along the cavity, i.e., the laser pulse is centered within the simulation window. Furthermore, due to the large amount of data produced per simulation timestep only a subset of all timesteps are actually saved to disk.

3.2.3 Data Overview

VORPAL uses two different kinds of output, one for storing snapshots of the entire simulation state at a particular point in time ("dumps") and the other being time histories of selected quantities. The dump data is both used for extracting simulation results, as well as for checkpoint/restarting operations often required in a timeshared environment. Dumps consist of field data, particle data and auxiliary state data, all written according to Vizschema [121], a self-describing data organization scheme for scientific data, designed to facilitate the visualization of its contained data.

The field data is defined on a computational grid. For each timestep separate datafiles are created for the electric field (with E_X, E_Y, E_Z) the magnetic field (with B_X, B_Y, B_Z), and plasma related variables such as density (Sum_rhoJ, J_X, J_Y, J_Z). Together, the field data forms a collection of scalar and vector variables describing the state of the plasma at a given time. From this data also information about the location and intensity of the laser pulse can be derived.

The analysis discussed in this thesis focuses mainly on the particle data. In contrast to the field data, the particle data is scattered data. For each timestep a separate particle datafile is written to disk describing the current state, i.e., location and momentum, of the macro-particles. Each macro-particle is represented as a vector of seven quantities, in the 2D case $(x, y, px, py, pz, id, wt)$ and eight in the 3D case (including z). The quantities x, y, and z are measured in meter (m) and describe the physical location of a particle. px, py, and pz are in $\frac{m}{s}$ (γv) and describe the momentum of a particle in x, y, and z direction respectively. wt describes the weight of each macro-particle defined by the number of electrons it represents. id is a unique identifier for each particle.

As a derived quantity, $x_{rel}(t) = x(t) - max(x(t)) + x_{slippage}$ is used in the analysis to describe the relative particle position in x direction within the simulation window. In the plasma the laser velocity v_{laser} is smaller than the speed of light c so that the laser pulse slips back in the simulation window which moves at c. The term $x_{slippage} = t * 0.155\mu m$ in the definition of x_{rel} corrects for the resulting slippage of particles over the course of the simulation for the parameters used here.

Due to the large amount of data stored per dump, scientists have to find a compromise between data storage limits and accuracy of the analysis, e.g., to be able to reconstruct meaningful particle trajectories from the stored data. In contrast to the actual simulation, the temporal resolution of the stored data may, therefore, be insufficient to accurately resolve wave frequencies but is selected to be sufficient to resolve bunching and dephasing of particle beams.

The size of LWFA simulations datasets varies between several gigabyte (GB) in the 2D case and usually several terabytes (TB) in the case of 3D simulations. The amount of data created by a single simulation run depends on various user-defined parameters, e.g., (i) the spatial resolution of the simulation (i.e., number of particles per cell and resolution of the simulation-grid), (ii) the temporal resolution of the data (i.e., how many of the simulated timesteps are written to file), and (iii) the size of the simulation window (i.e., the area around the laser pulse captured by the simulation). With increasing performance of supercomputers and decreasing cost for storing simulation results, researchers are able to perform more accurate simulations with higher temporal and spatial resolution. The size of simulation datasets is, therefore, expected to increase even further in the future. With decreasing simulation cost also the number of simulations a researcher can perform increases, resulting in the creation of ever larger databases of simulation data.

To achieve good performance, data analysis procedures often employ dedicated data reduction methods and/or focus on specific features of the data. In the case of LWFA simulation data only a fraction of all particles are accelerated to relevant energy levels and an even smaller number of particles are actually part of the particle beam(s) of interest. Being able to efficiently access the relevant data subsets is crucial to enable high-performance analysis.

To enable efficient identification and access of data subsets, the analysis methods described in this work make use of state-of-the-art data management using FastBit [51] (see also Section 2.6). The particle data is converted to H5Part and augmented with FastBit bitmap indices. To enable efficient access to the data, this work makes use of HDF5-FastQuery [122]; a data access API based on HDF5 [120][4], H5Part [123, 124][5] and FastBit [51].

[4]HDF5 (Hierarchical Data Format); is a self-describing machine-independent binary file format.

[5]H5Part is a data storage schema and API build on top of HDF5 with the goal to define a common file storage format. The goal of H5Part is to enable stable visualization and data analysis environments and interoperability

3.2.4 Data Analysis in Laser Wakefield Accelerator Physics

Knowledge discovery from large, complex simulation data is a challenging task. Visualization and statistical analysis are common tools to address this problem. A large number of analysis frameworks are available for this purpose, e.g., ROOT [125], PAW [126], AIDA [127], R [128] , IDL [129] OpenDX [130], VorpalView [131], ParaView [36], and VisIt [34]. While analysis based on statistics and visualization has proven very useful most of the available analysis are based on static views of the data. Animation of static plots is commonly used for investigation of the temporal evolution of the data. This type of analysis is, however, static in the sense that the user usually has no or only limited ability of dynamic interaction with the visualization. While visualization systems, such as VisIt and ParaView, support dedicated data selection mechanisms, their ability to define data queries interactively for extremely large particle accelerator simulations is limited. Also, none of the above mentioned tools addresses the specific problem of automatic analysis of particle beams in LWFA simulations.

Fonseca et al. [132] described recently a framework for particle tracing in the context of LWFA simulation based on the OSIRIS [118] framework. The size of the data prohibits saving the complete information of every simulated time step. To get high-quality particle traces they, therefore, execute the simulation twice. After the first simulation run a researcher manually defines the particle subset of interest and then re-executes the simulation to gather the data of the selected particle subset at a higher temporal resolution. Selection of the particles of interest is execute manually by selecting, e.g., the n most energetic particles at the last time step. Martins et al. applied these methods to investigate the ion dynamics and acceleration in relativistic shocks [133]. Analysis based on serial particle tracing procedures was also described by Tsung et al. [113] and Cameron Geddes [106]. In contrast to the efforts of Fonseca et al. and Martins et al. this work focuses on how to efficiently trace particles in a large collection of data and the automatic classification of particle bunches based on particle paths.

Recently methods for automating different parts of the data analysis process have received attention. Bagherjeiran et al. [134] presented a comprehensive report on applying graph-based techniques for orbit classification in plasma simulations. This approach targets the system dynamics in particle accelerator data in terms of particle orbits but does not address particle dynamics as a function of time or inspection of particle bunches. Love et al. [135] conducted an image space analysis of coherent structures in plasma simulations using a number of segmentation and region-growing techniques to isolate regions of interest in orbit plots. Hlína et al. [136] studied dynamic patterns and their velocities in thermal plasma jets via substraction and correlation

across project's software infrastructures.

analysis of succeeding images in a time series of CCD images recording the plasma-radiation. In contrast to the work presented in this thesis, Hlína et al. and Love et al. focus on structures of the plasma itself rather than the dynamics and behavior of individual or groups of particles.

Chapter 4

Visualization and Analysis of 3D Gene Expression Data

The recent development of methods for extracting precise measurements of spatial gene expression patterns from three-dimensional (3D) image data opens the way for new analyses of the complex gene regulatory networks controlling animal development. An integrated visualization and analysis framework is presented that supports user-guided data clustering to aid exploration of these new complex datasets. The interplay of data visualization and clustering-based data classification leads to improved visualization and enables a more detailed analysis than previously possible.

An interface between the visualization and MATLAB®[1] is introduced to enable bioinformatics researcher to more easily integrate their analysis with the visualization. Using this interface advanced analysis algorithms can be easily integrated with the visualization and made readily accessible for the target user.

As novel scientific contributions of this work this chapter discusses the:

(i) integration of data clustering and visualization into one framework;

(ii) application of data clustering to 3D gene expression data;

(iii) evaluation of the number of clusters *k* in the context of 3D gene expression clustering;

(iv) improvement of overall analysis quality via dedicated post-processing of clustering results based on visualization;

(v) integration of the visualization system PointCloudXplore (PCX) with MATLAB; and

[1]MATLAB is a registered trademark of The MathWorks Inc., 3 Apple Hill Drive Natick, MA 01760-2098, USA. Online at: `http://www.mathworks.com/`

(vi) as specific applications, the use of the proposed framework to: (a) objectively define spatial boundaries of gene expression patterns, (b) characterize the temporal variation of genes, and (c) to analyze how mRNA patterns are controlled by their regulatory transcription factors.

4.1 Problem

Understanding the control of embryo development is a fundamental question in biology. A cell's unique fate is determined by specific combinations of developmental regulatory factors that form part of complex genetic regulatory networks ultimately coordinating the expression of all genes. As a result, the developing embryo exhibits an extraordinarily complex set of spatial and temporal gene expression patterns. The basic structure of the genetic regulatory network is defined by the genome sequence. However, it is currently not possible adequately decipher this information or correctly predict how patterns of gene expression evolve.

The *Berkeley Drosophila Transcription Network Project* (BDNTP) is generating multiple complementary datasets to address these challenges using the early *Drosophila* developmental regulatory network as a model. These data sets include *in vitro* and *in vivo* DNA binding data for key transcriptional regulators and, of particular relevance to this work, 3D gene expression data that describes the spatial output of the network at cellular resolution for multiple time points [91, 94].

Many relevant questions can be addressed using these new 3D gene expression datasets [94, 99]. For some analyses, such as logic-based network models, it is helpful to have an objective description of the pattern of a gene at a particular time point, i.e., to define which cells do or do not express a gene. Analysis of the temporal dynamics of gene expression, i.e., how patterns change over time, is essential for gaining a deeper understanding of complex network interrelationships. Knowledge of the input and output of a network, i.e., the response of the gene expression network at time $t = t_{i+1}$ to the input of the expression levels of regulators at time $t = t_i$, is paramount to identifying regulatory interactions.

To address these and other challenges, a flexible visualization and analysis tool that allows for interactive exploration of the data is needed. Since *drosophila melanogaster* has been used as a model for genetic research for decades, there exists a large accumulated body of knowledge about it. A tool designed for the analysis of 3D gene expression data must therefore allow researchers to incorporate this existing knowledge in the analysis, for example by providing ways to modify analysis results, and thus the visualization, accordingly. The tool must also capture the biological context of the embryo and allow different subsets of the data (cells or gene expression patterns) to be examined.

The work described in this chapter is based on the 3D gene expression visualization framework PointCloudXplore (PCX). Section 4.2 introduces PCX and describes its many features. Afterwards, Section 4.3 provides an overview of the proposed framework for knowledge discovery from 3D gene expression data based on the integration of data clustering (Section 4.4 and 4.5) and MATLAB (see Section 4.6) with PCX. This chapter concludes with a presentation of results and applications of the proposed analysis framework (see Section 4.7).

4.2 State of the Art: Visual Analysis of 3D Gene Expression Data

PointCloudXplore (PCX)[2] is a visualization system specifically developed for the analysis of 3D gene expression data [157, 159][137]. PCX is the standard visual analysis software of the BDTNP and integrates physical and abstract data views in a common framework using the established concept of brushing and linking (see Section 2.3). In the context of PCX the, with respect to the application, more intuitive term cell selection is used instead of brushing. Accordingly a brush —i.e., an entity used for cell selection— is referred to as cell selector. Physical views use a model of the embryo to visualize spatial gene expression patterns (see Section 4.2.1). In abstract views, physical cell positions are ignored and expression levels for multiple genes are plotted with respect to each other using scatter plots or parallel coordinates (see Section 4.2.2).

The different physical and abstract views PCX supports are useful in their own right and can be used individually to mine data sets for new information. However, it is often desirable to correlate information shown in different views. Selecting cells of interest can be executed in any view in PCX, while depending on the view, different data properties are employed in the select process. User-defined cell selections are then stored and managed in a central cell selector management system (see Section 4.2.3). Since all views have access to the same set of cell selectors, features of interest can be defined in any one view and then further analyzed in any other view. The most common way to visualize cell selectors in PCX is to use a consistent color mapping. Depending on the current view, additional functions for highlighting cell selectors are available, such as cell selector-bands in 2D parallel coordinates.

Figure 4.1 shows a snapshot of the GUI of PCX. The main window is split into two main

[2]PointCloudXplore is online available at: `http://bdtnp.lbl.gov/Fly-Net/bioimaging.jsp?w=pcx`

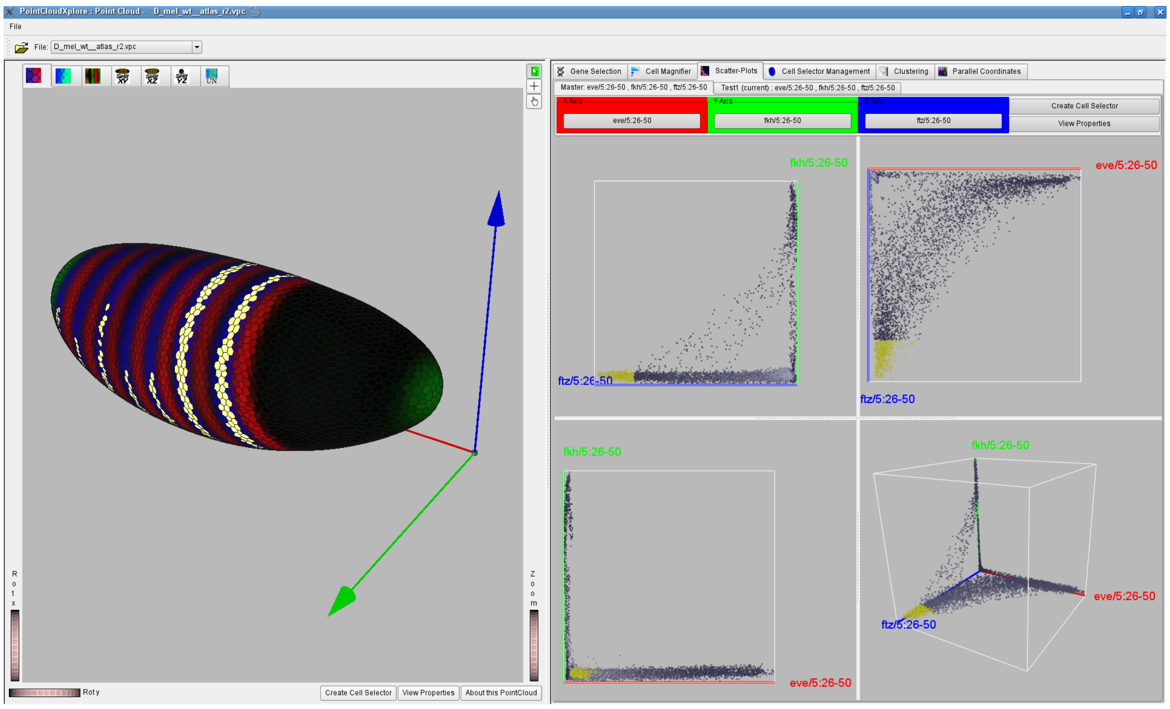


Figure 4.1: Main user interface of PointCloudXplore showing a physical view of the embryo (left) and scatter plot view (right). Both views show the genes *eve* (red) and *ftz* (blue) at stage 5:26-50% of embryo development as well as an user-defined cell selection (yellow).

areas: The left part contains all physical views of the embryo (e.g., a Cell View in Figure 4.1) and the right part all abstract views (e.g., a scatter plot in Figure 4.1) as well as additional user controls such as the cell selector management. These two parts of the window can be resized within the main window by moving a central split bar, allowing the physical- or abstract views to occupy as much screen space as desired by a user. Controls and abstract views in the right part are arranged in a series of detachable tabs such that a user can switch between different abstract views or detach them from the main window and show them side-by-side.

4.2.1 Physical Views

Physical views use a 3D embryo model, or different 2D projections of this 3D model, to convey a sense of the spatial distribution of gene expression on the blastoderm. There are three physical views in PointCloudXplore (PCX): (i) *3D Views* (see Section 4.2.1.1), (ii) *Orthographic View*, and (iii) *Unrolled View* (see Section 4.2.1.2). Each view has its strengths and weaknesses in presenting aspects of gene expression patterns. The 3D Views provide the most spatially flexible representation of the embryo. Orthographic Views simulate the 2D views of embryos that most biologists are used to. Finally, the Unrolled View shows a 2D map of all blastoderm cells providing an overview of the expression in the entire embryo.

All physical views use color intensity to show expression levels analogous to the way staining was used to reveal gene expression levels in the original embryo. Color is also used to display cell selections, i.e., to highlight which cells are selected by a set of user defined cell selectors.

In physical views, selection is performed by painting patterns on the embryo. In this process the user subsequently adds or removes cells from a cell selector by drawing over the cells of interest using the mouse-cursor.

In the 2D views the third dimension is freed-up and can be used for visualization of expression values. Using so call expression surfaces (see Section 4.2.1.3) the user can analyze gene expression pattern in much more detail than possible when just using color. For more details about physical views see also [157, 159, 162] and [137].

4.2.1.1 Three-dimensional Embryo Views

PointCloudXplore's 3D Embryo Views utilize a 3D model of the embryo, which a user can rotate, pan, and scale to obtain an overview of the entire embryo (Figure 4.2 left). Cells can be represented in three ways in the 3D Views: i) as spherical cells in the so called *Sphere View* (not shown), ii) as points in the approximated Delauny triangulation of the blastoderm surface in the so called *Smooth View* (not shown), or as polygonal cells in the so called *Cell View* (see 4.1 left or 4.2 left). In the following the Cell View is used as standard 3D visualization of the embryo.

In the Cell View, a surface composed of polygonal faces, each of which corresponds to a detected nucleus, represents the embryo. These polygons form an approximate Voronoi tessellation of the blastoderm surface and have a visual appearance similar to that of cells. The blastoderm surface is assumed to be a two-manifold (i.e., a locally flat surface) and polygon size depends on the distribution of cells on the embryo blastoderm.

4.2.1.2 Two-dimensional Embryo Views

The different 2D embryo views are projections of the Cell View. Using these 2D projections a user can gain an overview of all cells of the embryo without the need for rotation.

Traditionally, biologists have studied expression patterns from photomicrographic images from defined views, e.g., ventral, dorsal, or lateral view, of the embryo. Orthographic Views simulate these familiar views by splitting the embryo along a body axes and then projecting the halves orthographically (not shown here).

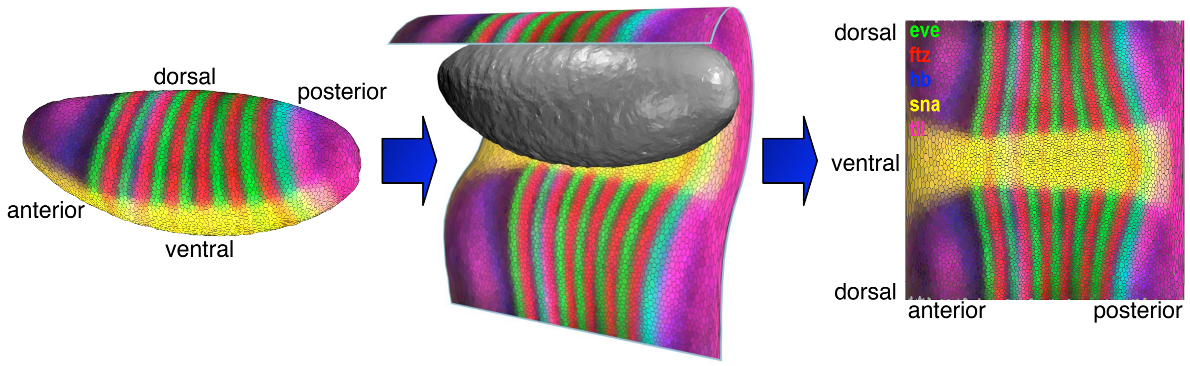


Figure 4.2: The Unrolled View uses cylindrical projection to map the entire embryo to a plane.

While Orthographic Views provide an overview of the entire embryo the resulting views are split into two sub-views and information at the edges of the projection is compressed. To alleviate these shortcomings, the Unrolled View maps the entire *Drosophila* embryo continuously to a plane using cylindrical projection [94], [157, 159, 162] (see Figure 4.2). Even though the 2D cylindrical projection distorts spatial relationships (especially in the termini), the Unrolled View provides and overview of all cells of the embryo. In the following the Unrolled View is used as standard 2D view of the embryo.

4.2.1.3 Expression Surfaces

Projecting the embryo to a plane has the advantage of freeing the z dimension up for displaying additional information. In PointCloudXplore (PCX), the free z dimension can be used to display gene expression values as surface plots, termed *Gene Expression Surfaces* [157], that support simpler quantitative analysis of gene expression data. Individual Expression Surfaces display data for one gene over either the Orthographic or the Unrolled Views. The xy-positions of Expression Surface points are determined by the positions of cells in the underlying view, whereas the height of an Expression Surface is determined by the expression values measured for the gene it represents.

Figure 4.3(a) shows the quantitative expression levels of the mRNA expression pattern of the transcription factor *eve* demonstrating how its expression level change along the anterior/posterior (A/P) as well as dorsal/ventral (D/V) axis. Multiple surfaces allow users to compare the quantitative relationship between genes. Figure 4.3(b), e.g., shows the quantitative relationship between the mRNA expression patterns of the transcription factors *ftz* (green) and *eve* (orange) illustrating that these genes do not only exhibit excluding alternating patterns along the A/P body axis but also differ in their variation along the D/V-axis [94].

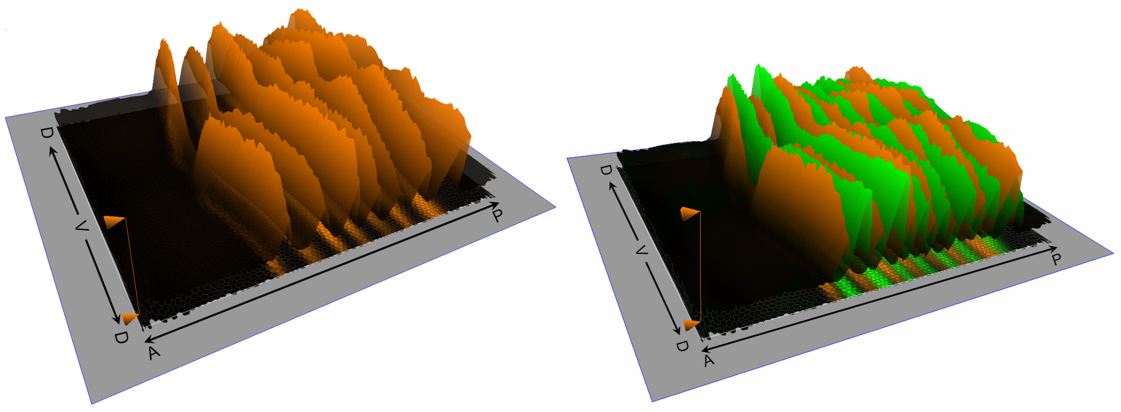


Figure 4.3: a) Surface showing the quantitative gene expression levels for *eve* (orange). b) An additional surface shows expressions levels for *ftz* (green). The direction of the anterior/posterior (AP) axis and the dorsal/ventral (DV) circumference are indicated.

4.2.2 Abstract Views

To explore relationships between gene expression's independently of their spatial context, PointCloudXplore (PCX) uses information visualization techniques in *gene expression space*. This is particularly useful for characterizing regulatory relationships between genes. PCX supports three main abstract views: i) scatter plots (see Section 4.2.2.1), ii) Cell Magnifier (see Section 4.2.2.2), and iii) parallel coordinates (see Section 4.2.2.3).

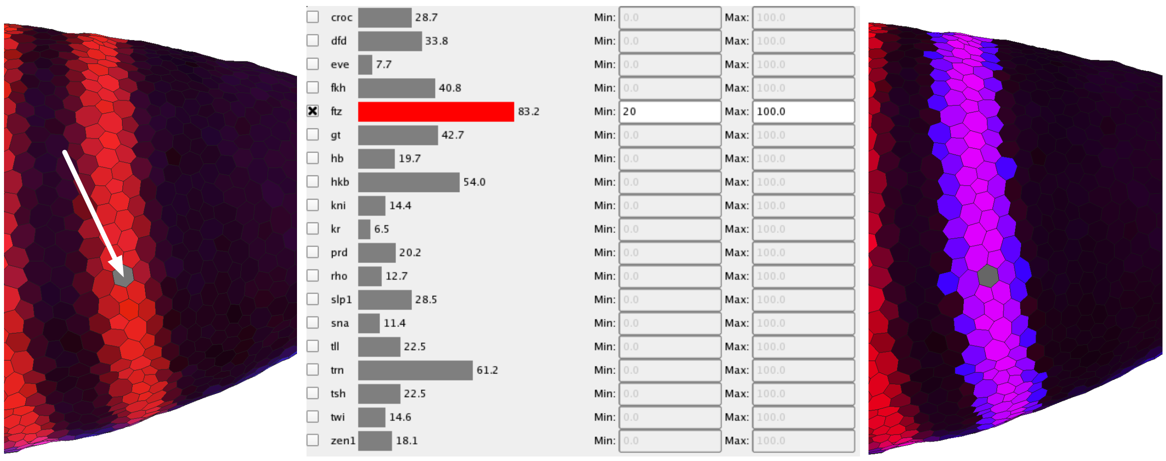
4.2.2.1 Scatter Plots

Scatter plots are conceptually the simplest way to visualize relationships in gene expression space (see Figure 4.1 right). Three genes are selected and mapped to the three axes of a Cartesian coordinate system where each axis represents one gene's expression level ranging from no expression at the origin to maximum relative expression. Each cell in the embryo is represented by a single point in the 3D scatter plot where the point's location is specified based on the relative gene expression levels of the according cell. The user can select cells of interest within a scatter plot by moving and resizing an axis aligned box within the plot. In PointCloudXplore (PCX) a 3D scatter plot is augmented with a set of 2D Scatter Plots that show expression relationships between the three possible pairs of genes in the 3D plot. The 2D plots provide a standard view on the data and facilitate cell selection (see Figure 4.1 right).

4.2.2.2 Cell Magnifier

Unlike the other physical and abstract Views available in PointCloudXplore (PCX), the Cell Magnifier (see Figure4.4, center panel) concentrates not on comparing gene expression values in different cells but on comparing expression values in just one cell. Gene expression values are visualized using a bar graph with one bar for each gene, colored according to the user defined stain colors. Since exact expression values can only be roughly estimated from bar size, the exact measured gene expression value is also displayed beside each bar. The cell to be displayed in the Cell Magnifier can be selected in any physical view and is highlighted by graying it out (arrowed in Figure 4.4, left panel).

To ease selection of regions in physical space with specific expression properties, PCX provides *Seed Cell Selection*, which uses the cell selected using the Cell Magnifier as seed point (see Figure 4.4). In addition to the seed cell the user specifies in the Cell Magnifier a set of expression levels that should be considered in defining the region of the embryo. Seed Cell Selection then uses a flood fill method [138] to identify all cells in a contiguous region whose expression levels lie within the specified expression range(s) (see Figure 4.4).



☐	croc	28.7	Min: 0.0	Max: 100.0
☐	dfd	33.8	Min: 0.0	Max: 100.0
☐	eve	7.7	Min: 0.0	Max: 100.0
☐	fkh	40.8	Min: 0.0	Max: 100.0
☒	ftz	83.2	Min: 20	Max: 100.0
☐	gt	42.7	Min: 0.0	Max: 100.0
☐	hb	19.7	Min: 0.0	Max: 100.0
☐	hkb	54.0	Min: 0.0	Max: 100.0
☐	kni	14.4	Min: 0.0	Max: 100.0
☐	kr	6.5	Min: 0.0	Max: 100.0
☐	prd	20.2	Min: 0.0	Max: 100.0
☐	rho	12.7	Min: 0.0	Max: 100.0
☐	slp1	28.5	Min: 0.0	Max: 100.0
☐	sna	11.4	Min: 0.0	Max: 100.0
☐	tll	22.5	Min: 0.0	Max: 100.0
☐	trn	61.2	Min: 0.0	Max: 100.0
☐	tsh	22.5	Min: 0.0	Max: 100.0
☐	twi	14.6	Min: 0.0	Max: 100.0
☐	zen1	18.1	Min: 0.0	Max: 100.0

Figure 4.4: Using the Cell Magnifier and Seed Cell Selection to select all cells of a *ftz* stripe.

4.2.2.3 Parallel Coordinates

PointCloudXplore (PCX) uses parallel coordinates to facilitate the visualization of relationships between many genes' expression. Using two sliders attached to each parallel axis a user can define cells of interest in parallel coordinates via multi-dimensional thresholding. To effectively display cell selections in parallel coordinates, PCX provides several extensions to standard parallel coordinates such as Cell Selector Bands (see Figure 4.5a).

PCX employs 3D parallel coordinates to reveal also spatial relationships between gene expression patterns in parallel coordinates. In 3D parallel coordinates the coordinate axes are extruded into the third dimension so that each axis effectively defines a 2D scatter plot (see Figure 4.5b). Data lines are ordered along the third dimension from back-to-front according to either: i) cell positions along the AP axis or the DV circumference, or ii) the expression of a selected gene. Along any given data line, the positional information is constant, such that data lines do not intersect each other in this third dimension. For a detailed description of PCX and parallel coordinates see [157] and [137].

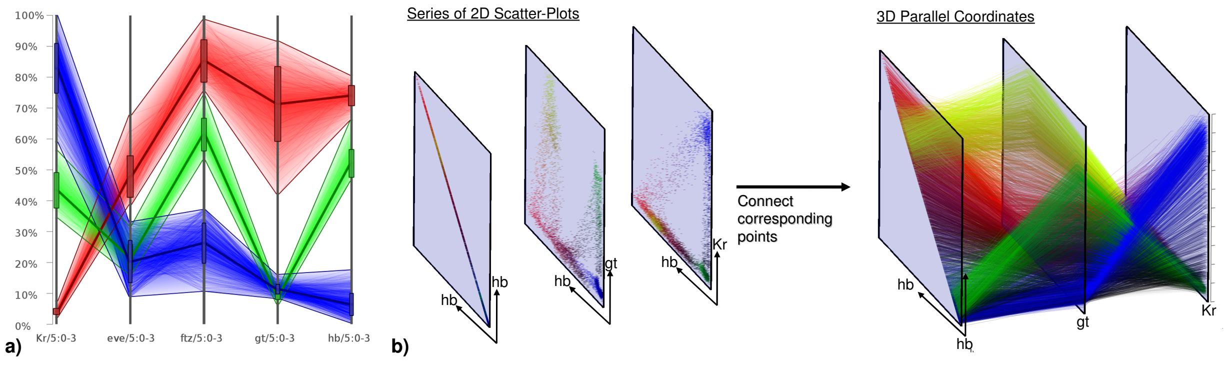


Figure 4.5: a) Parallel coordinates visualization of three cell selectors using cell selector bands and average curve lines. b) Illustration of 3D parallel coordinates.

4.2.3 Cell Selector Management

All views available in PointCloudXplore (PCX) are linked via a central *cell selector management system*. Cell selectors of any kind can be accessed here in a unified way and common cell selector properties, such as color, can be defined. To support dedicated selection mechanisms for different views PCX provides a set of different cell selectors: i) *DATA* cell selectors, used in scatter plots, ii) *POSITION* cell selectors, used in the physicale views as well as for Seed Cell Selection, and iii) *PATTERN* cell selectors, used in parallel coordinates. All cell selector types are derived from a common base class so that all cell selectors can be displayed in any view.

The central cell selector management allows one not only to perform basic management operations, but also supports combining cell selectors using logic operations, such as, AND, OR, and NOT. Thus, for example, cell selectors defined in a scatter plot can be combined with cell selectors defined by drawing on the embryo or via seed cell selection, making it possible to define higher-order cell selections within gene expression space. Logical combinations of cell selections are implemented via so called *logical cell selectors.* Logical cell selectors are a subset of cell selectors that are not manually defined by the user, but are computed by a logical operator using other cell selectors as input. PCX supports three types logical cell selectors: i) AND defines the intersection of two cell selectors, ii) OR defines the union of two cell selectors, and iii) NOT inverts the selection of a cell selector. An example illustrating the use of logical cell selectors is provided later in Section 4.7.1. For more details about the cell selector management in PCX see the PCX user manual [137].

4.3 Overview

While visualization is a powerful approach to gain deeper insights into complex data sets, it is limited in this case. The intricate and often subtle nature of 3D gene expression data makes visual detection of all existing features very difficult. For example, a typical feature of interest would be various groups of cells behaving similarly with respect to the expression of several genes. A human's eye and mind, however, cannot readily compute relative concentrations of gene products.

Data clustering has already proven to be very powerful at revealing details from conceptually simpler forms of expression data, such as that from microarray experiments, that are not easily detected visually in raw data. Appropriately defining clustering parameters, such as the number of clusters, as well as validation and interpretation of clustering results, is a complicated endeavor.

To overcome these difficulties in both visual analysis and data clustering, this work adapts data clustering for 3D gene expression analysis by integrating it into PointCloudXplore (PCX). Figure 4.6 provides an overview of the framework for knowledge discovery from 3D gene expression data based on PCX. Data clustering facilitates the detection of groups of cells showing similar expression behavior. Automatically created clusters can be interpreted as a special types of cell selections. Clusters are managed in PCX by the central cell selector management system with additional management options specific to the management of clusters. This allows all views available in PCX to access and visualize the clusters created by the automatic clustering procedure (see Section 4.5.1 for details).

In the field of bioinformatics a variety of specific algorithms for the analysis of gene expression data are available. A new interface between PCX and MATLAB is introduced to make advanced data analysis tools more readily accessible to the average biologists and to allow bioin-

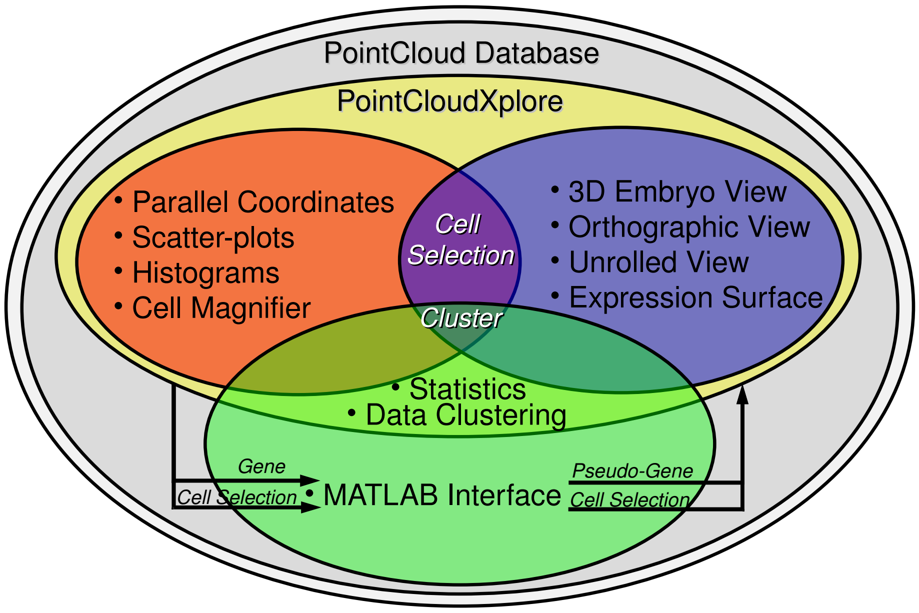


Figure 4.6: Analysis framework for knowledge discovery from 3D gene expression data.

formatics experts to more easily validate and deploy their analysis. MATLAB is a technical computing language and environment for algorithm development, numeric computation, as well as data analysis and visualization. MATLAB is widely used in the bioinformatics research community and provides, e.g., also a dedicated bioinformatics toolbox[3]. With its interface to MATLAB, PCX can directly call selected analysis functions implemented in MATLAB, transfer data, such as, gene expression values, cell locations, or cell selection to MATLAB, and import analysis results in the form of derived-genes or cell selections.

The proposed framework based on the integration of data analysis and visualization enables a more detailed analysis of 3D gene expression data than previously possible. This chapter discusses as specific application examples the use of the proposed framework to: i) objectively define spatial pattern boundaries (see Section 4.7.2), ii) analyze the temporal variation of genes (see Section 4.7.3 and 4.7.6), and iii) analyze how mRNA patterns are controlled by their regulatory transcription factors (see Section 4.7.1, 4.7.4 and 4.7.5).

Within the scope of the proposed framework for knowledge discovery from 3D gene expression data this work presents the following novel scientific contributions:

(i) To enable automatic classification of cells based on expression this work adapts data clustering to 3D gene expression data (see Section 4.4.1).

[3]See online at: `http://www.mathworks.com/products/bioinfo/`

(ii) Section 4.4.2 introduces a novel approach for evaluation of the number of clusters *k* in the context of 3D gene expression clustering.

(iii) To enable detailed analysis of 3D gene expression data, this work describes the integration of data clustering and visualization into one framework (see Section 4.5).

(iv) As part of this framework, Section 4.5.2.1 presents dedicated post-processing methods for improvement of clustering results based on visualization.

(v) A novel interface between MATLAB and the visualization system PointCloudXplore (PCX) makes advanced data analysis more readily accessible to PCX users (see Section 4.6).

(vi) Finally, this work describes several specific applications showing the use of the proposed framework to: (a) discretize gene expression patterns, (b) analyze the temporal variation of genes, and (c) investigate how the pattern of a gene is regulated (see Section 4.7).

In the following, first, the application of data clustering to 3D gene expression is described (see Section 4.4). Afterwards, the integration of data clustering and MATLAB with PCX are described in detail in Sections 4.5 and 4.6 respectively. Section 4.7 then provides an overview of results and applications of the proposed framework for knowledge discovery from 3D gene expression data.

4.4 Clustering 3D Gene Expression

This section addresses the questions: i) How to cluster 3D gene expression data? (see Section 4.4.1), and ii) How to define the number of clusters *k*? (see Section 4.4.2). Afterwards, the integration of data clustering into PointCloudXplore (PCX) and the workflow of a 3D gene expression clustering experiment are described in Section 4.5.

4.4.1 Cluster Analysis of 3D Gene Expression Patterns

3D gene expression data can be described as a matrix where each row represents one cell and each column one cell attribute, i.e., the expression of a gene at a specific time point or the x, y, z position of the cells in physical space. Note that this form of data is quite different from that of gene expression microarray matrices, where each row represents a gene and each column represents expression under a different experimental condition, and spatial relationships are meaningless. The fact that 3D gene expression data can be expressed as an expression matrix allows standard clustering algorithms to be applied here to classify cells into groups (the

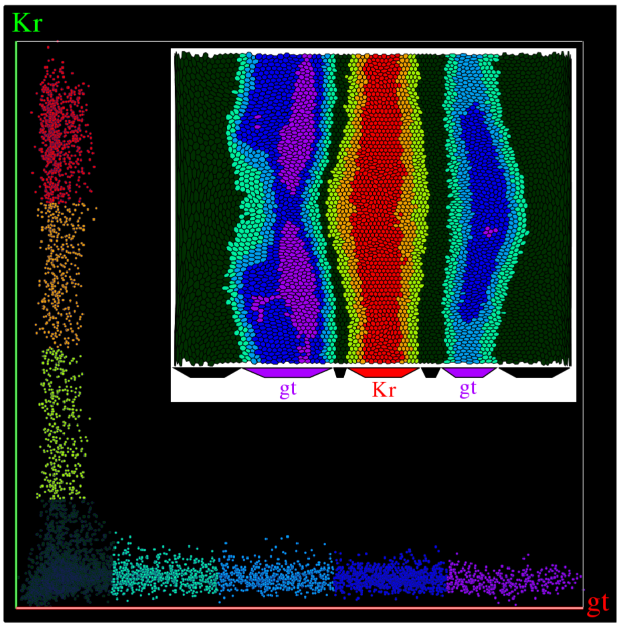


Figure 4.7: An example clustering of *giant*(*gt*) and Krüppel (*Kr*) using k-means clustering and Euclidean distances with $k = 8$. In the scatter plot, the structure of the clusters is shown in expression space, while the Unrolled View reveals spatial structures formed by the different clusters.

clusters) of similar expression behavior. To account for the spatial relationships between cells, dedicated data selection and cluster post-processing approaches are introduced, described later in Sections 4.5.1.1 and 4.5.2.1.

To implement clustering operations in PointCloudXplore (PCX), portions of the open source clustering library *Cluster 3.0* [59] are used. Data clustering is directly integrated into PCX and a dedicated GUI provides access to the data clustering functionality and allows management of clustering results (see Section 4.5.1). Clustering algorithms currently available in PCX include the most commonly used methods for microarray gene expression data analysis, such as k-means, k-median, and k-medoid clustering, as well as several hierarchical clustering algorithms, and self-organizing maps (SOMs) [59, 76, 77, 64]. All these clustering algorithms require an appropriate distance function in order to define similarity between cells. Within PCX the most common metrics for defining distances in gene expression space are made available: i) Euclidean distance, ii) city-block-distance, and iii) several derivatives of the Pearson correlation [59].

Some clustering algorithms require additional parameters, such as the number of clusters k, to be specified by the user. In the context of 3D gene expression data there exists in general not a single near-optimal value for k, but one rather finds a number of valid values, each representing a different level of detail. This behavior is due to the fact that quantitatively different expression levels of a gene may lead to multiple different outputs of the underlying genetic regulatory network. It is therefore valid to subdivide elongated structures formed in gene expression space into several sub-clusters.

For example, considering early-stage *giant* (*gt*) and *Krüppel* (*Kr*), which are expressed in spatially non-overlapping patterns, leading to formation of an L-shaped scatter plot (see Figure 4.7). Even though one could interpret this structure as one cluster — possibly indicating a *NOT* relationship between *gt* and *Kr* — it is also valid to subdivide this structure into, e.g., eight clusters, resulting in one cluster representing background expression, a three-level description of the pattern of *Kr*, and a four-level description of the *gt* pattern.

The choice of k depends on the level of detail required by the user. Therefore, an interactive process is used to define k based on visualization (see Section 4.4.2). The spatial structure formed by the cells selected by clusters, cluster statistics, and standard data visualizations provides a way to decide if the number of clusters should be increased or reduced. Depending on the characteristic spatial patterns of genes, the cells included in a cluster often define some coherent spatial pattern. Thus, the presence of clusters that show high spatial scattering may be an indication that the chosen k was too large. To assist in this evaluation process, a novel dedicated cluster quality measure is presented in Section4.4.2 indicating the physical scattering of clustering results along with a function for suggesting a good initial k.

4.4.2 Identifying Good Values for *k*, the Number of Clusters

Many clustering algorithms, such as k-means, require the user to specify as an input parameter the target number of clusters, k. The quality of clustering results often depends on a proper choice of k. Unless users have *a priori* knowledge concerning the number of clusters present in the data, it is helpful for the user that the software offers a reasonable, initial value for k. Different approaches for finding an approximately optimal k have been proposed. In the context of 3D gene expression clustering, those methods based on internal cluster measures are more appropriate [65, 74]. The objective of the presented approach, as described below, is to provide the user with assistance in the interactive search for a good k as opposed to trying to automatically compute the optimal value of k.

Even though internal cluster quality measures may be useful in this context, none of the measures described earlier in Section 2.7.2.2 take the specific characteristics of 3D gene expression data into account. Genes are often expressed in characteristic spatial patterns, and one therefore expects the derived clusters to be spatially compact. The presence of computed clusters with high spatial scattering typically suggest that the value of k was too large. Information about the physical cell position is usually not used in the clustering process hence, spatial compactness is a criterion available as an independent measure for clustering quality. As discussed below, spatial cluster scattering can also serve as a measure to indicate a series of adequate values of k. Combining spatial cluster scattering and the clustering error in expression space yields a method to identify a good initial value for k that accurately reflects the structures present in the data, but with relatively low spatial scattering.

As an objective measure for the relative spatial scattering of a clustering result the function:

$$\varepsilon_{\text{scatter}}(k) = \frac{\sum_{i=1}^{k} R_1(i)}{\sum_{i=1}^{k} R_\infty(i)} \tag{4.1}$$

is used. $R_s(i)$ (with $s > 0$) is the number of spatially independent components of cluster i consisting of at most s cells. $R_1(i)$ thus defines all single cell regions in cluster i. $R_\infty(i)$ is the total number of spatially independent regions in cluster i. $\varepsilon_{\text{scatter}} \in [0,1]$ is independent of the clustering algorithm, usually has discontinuities, and shows a larger variation for smaller values of k than for large values of k. Local minima of $\varepsilon_{\text{scatter}}$ indicate values of k for which clusters are relatively compact, and thus indicate a series of appropriate values of k. In the context of 3D gene expression data, clustering errors introduced by single cells isolated in physical space are quite common and the proposed construct for $\varepsilon_{\text{scatter}}$ performs well. An alternative approach might work better when these cluster outliers consist of small groups of cells. One approach might be a less sensitive weighted cascade measure that also accounts for larger regions as potential scatter, such as

$$\varepsilon_{\text{sc}}(k) = \frac{\sum_{s=1}^{p}(\frac{1}{s} * \sum_{i=1}^{k} R_s(i))}{\sum_{s=1}^{p}(\frac{1}{s} * \sum_{i=1}^{k} R_\infty(i))}, \tag{4.2}$$

with $p > 0$ being much smaller than the number of cells.

To evaluate the clustering error in expression space ε_{exp}, the average distance, in expression space, of a cell to the center of the cluster it belongs to,

$$\varepsilon_{\text{exp}}(k) = \frac{1}{n}\sum_{i=1}^{n} dist(center(c_i), c_i), \tag{4.3}$$

is used, where n is the number of clustered cells, c_i is the i^{th} cell, $center(c_i)$ is the center of the cluster to which c_i belongs, and $dist(\cdot,\cdot)$ is the distance operator used in the clustering process.

$\varepsilon_{\text{scatter}}$ and ε_{exp} are computed for $2 \leq k \leq m$ by executing a series of clusterings, with m being the first value where $\varepsilon_{\text{scatter}} > 50\%$. If the pattern of only one gene is used in the clus-

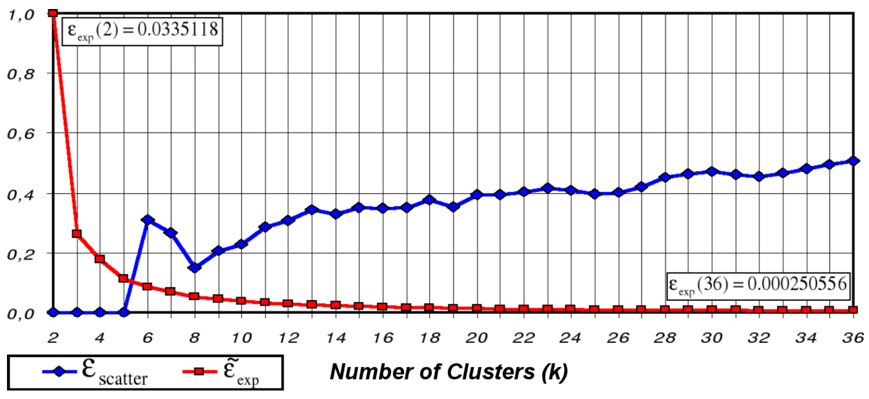


Figure 4.8: Cluster evaluation functions $\tilde{\varepsilon}_{\text{exp}}$ (red) and $\varepsilon_{\text{scatter}}$ (blue) for the clustering of *gt* and *Kr*, with $w = 5$ and $m = 36$. The suggested value for k is eight as shown in Figure 4.7.

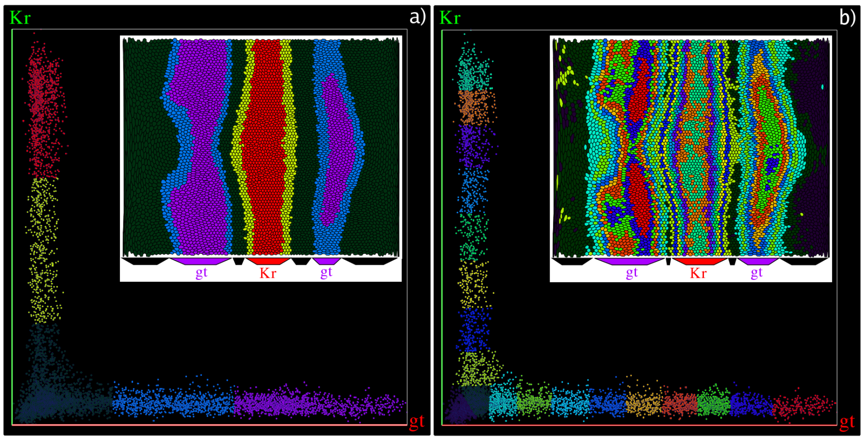


Figure 4.9: The patterns of *gt* and *Kr* are classified using k-means clustering, as in Figure 4.7, but with a) $k = 5$ and b) $k = 19$. One can see that the suggested $k = 8$ provides a compromise between a high level description as shown in a) and a detailed description as shown in b).

tering, $\varepsilon_{\text{scatter}} > 60\%$ is used as termination criterion instead because variations in background expression have a stronger impact on the cluster analysis, and because more complex structures are possible when multiple genes with spatially overlapping patterns are clustered. Using these thresholds for $\varepsilon_{\text{scatter}}$, ensures that the evaluation procedure iterates over all potentially useful clusterings and does not terminate prematurely. $k = 2$ serves as starting point because it represents the first potentially useful clustering. Furthermore, considering the relatively large value of $\varepsilon_{\text{exp}}(1)$, starting at $k = 1$ would result in a suggested value for k that is too small.

To identify a value w for k for which the error in expression space is sufficiently low to well characterize the data, the algorithm identifies the first k for which the decrease in ε_{exp} is lower than the average decrease

$$\bar{\Delta}_{\varepsilon_{\exp}} = \tfrac{\varepsilon_{\exp}(2)-\varepsilon_{\exp}(m)}{m-2}. \tag{4.4}$$

Alternatively w could also be defined as the k that corresponds to the point of the $\varepsilon_{\exp}$ evaluation curve that is furthest from the line defined by $\varepsilon_{\exp}(2)$ and $\varepsilon_{\exp}(m)$ [139]. While the first approach tries to find the k for which the expression error has sufficiently decreased, the second approach tries to identify the so-called *knee* of the $\varepsilon_{\exp}$ evaluation curve. Both methods depend on m, but this dependency is well behaved, i.e., with increasing m the suggested w changes slowly and continuously. During the research and development of this work, both methods seem to work equally well.

Based on $\varepsilon_{\exp}$, $\varepsilon_{\text{scatter}}$, and w, the following algorithm then identifies a good initial $k > w$ that also results in a relatively low physical scattering:

```
$k = w + 1$
$l = k$
for $i \leftarrow l$ to $m$
    do
        if ($\varepsilon_{\text{scatter}}(i) < \varepsilon_{\text{scatter}}(l) + t$)
        then
            $k = i$
            if ($\varepsilon_{\text{scatter}}(i) < \varepsilon_{\text{scatter}}(l)$)
                then $l = i$
```

Initially, k is set to $w+1$, which is the lowest value that results in a sufficiently low expression error. Then, the algorithm tries to optimize the expression error as well as the physical scattering by searching for a $k > w$ that also results in a relatively low physical scattering. A threshold of $t = 4\%$ is used — determined through empirical testing — to restrict the maximal allowed increase in $\varepsilon_{\text{scatter}}$ with respect to l, i.e., the k with the lowest relative physical scattering visited so far. Since $\varepsilon_{\exp}$ decreases with increasing values of k, the error in expression space for the suggested value for k is guaranteed to be smaller than $\varepsilon_{\exp}(w)$.

Alternatively, one can also view the problem of finding a good initial k as an optimization problem by looking for the k that minimizes

$$\varepsilon_{\text{total}}(k) = \mid \breve{\varepsilon}_{\exp}(k) - \tfrac{1}{w}\breve{\varepsilon}_{\text{scatter}}(k) \mid, \tag{4.5}$$

where both $\varepsilon_{\exp}$ and $\varepsilon_{\text{scatter}}$ are normalized. Conceptually, the first approach is more intuitive, does not require normalization of the evaluation functions, and will always suggest a minimum of $\varepsilon_{\text{scatter}}$ if an adequate local minimum exists. Using $\varepsilon_{\text{total}}$ for finding a good initial k has the advantage that it does not rely on a threshold t. Furthermore, it may result in a more reliable suggestion in cases where $\varepsilon_{\text{scatter}}$ is degenerate since $\varepsilon_{\text{total}}$ does not directly rely on the notion that the physical scattering increases with increasing values of k. In practice both approaches have shown to be useful.

Beginning with an initial, suggested value of k, the user can determine the best-suited k based on the information from the cluster evaluation and previews of the different evaluated clustering results using an Unrolled View. A more detailed description of the interface for evaluation of the number of clusters k is provided later in Section 4.5.1.2. Even though the initial, suggested value of k may not always be optimal, our testing has revealed there is value in providing a reasonable value or range of values for k.

Figure 4.8 shows the cluster evaluation functions for the clustering of *gt* and *Kr*. To provide an overview of both functions in one plot, all cluster number evaluation plots show $\varepsilon_{\text{scatter}}$ along with

$$\tilde{\varepsilon}_{\text{exp}}(k) = \frac{\varepsilon_{\text{exp}}(k)}{\varepsilon_{\text{exp}}(2)}. \tag{4.6}$$

The suggested value for k is eight, which is also a strong local minimum of $\varepsilon_{\text{scatter}}$. The corresponding clustering result for $k = 8$ is shown in Figure 4.7. Figure 4.9 shows two additional example classifications of *gt* and *Kr* using $k = 5$ and $k = 19$. $k = 5$ is the highest level for which $\varepsilon_{\text{scatter}} = 0$ and $k = 19$ is a local minimum of $\varepsilon_{\text{scatter}}$ ($\varepsilon_{\text{scatter}}(19) \approx 35.29\%$) close to the middle of the range. As this example illustrates, the suggested level of $k = 8$ provides a good compromise between high-level and low-level descriptions of the patterns. The value of k that is best suited to investigate a biological question depends to a large degree on user requirements. Further example usages of $\varepsilon_{\text{scatter}}$ and ε_{exp} are presented in Sections 4.7.2 and 4.7.3.

ε_{exp} and $\varepsilon_{\text{scatter}}$ are global cluster quality measurement functions in the sense that the clustering quality is evaluated based upon the entire dataset (in this case, all classified cells). Global error measures might not be appropriate if the user performs a clustering of a larger number of cells but is interested only in a small subset of clusters defining some local feature of interest.

4.5 Integrating Data Clustering and Visualization for the Analysis of 3D Gene Expression Data

The processing pipeline offered by PointCloudXplore (PCX) consists of two main interconnected components, visualization and data clustering, linked via the central cell selector and cluster management (see Figure 4.10). Visualization provides the ability to explore the data, to determine appropriate parameters for the clustering, to validate and analyze clustering results, and to

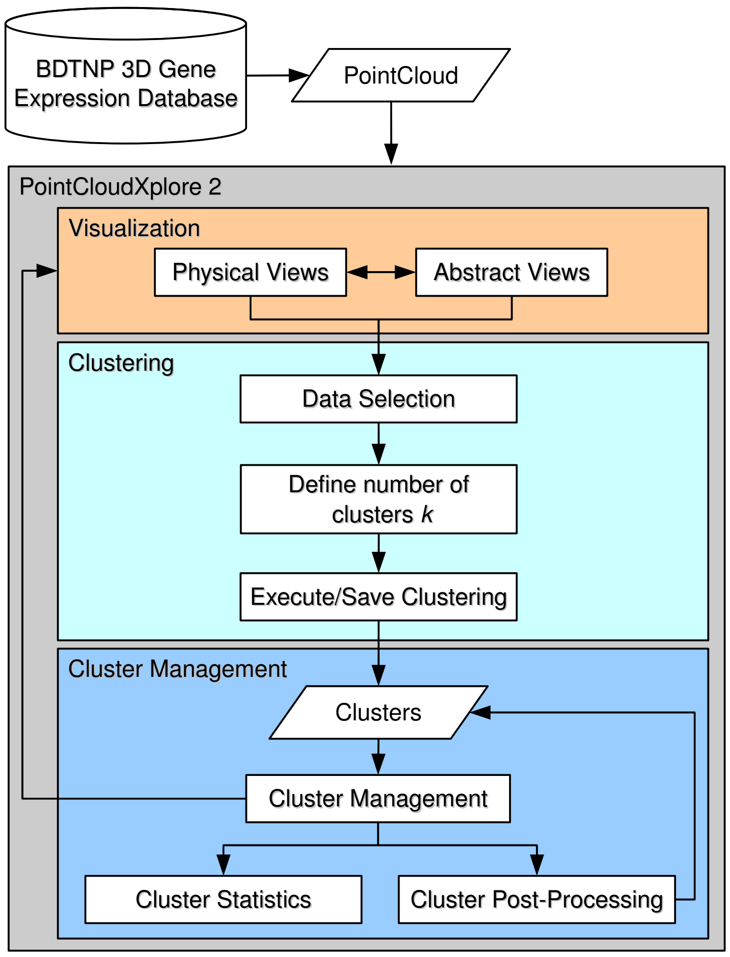


Figure 4.10: The data clustering and visualization pipeline. The pipeline illustrates the main components of PointCloudXplore (PCX) as well as the workflow of a 3D gene expression clustering experiment.

modify clustering results using several dedicated cluster post-processing techniques. Clustering provides ways for automatic identification of data features by classifying cells into groups (the clusters) based on similarity of their gene expression profiles. Like a manually created cell selector, an automatically created cluster defines a subset of cells in the embryo and can, therefore, be stored and visualized in the same way as cell selectors. By highlighting clusters in the visualization, analysis and comparison of specific data features becomes possible, enabling a more detailed and focused analysis of the data.

Figure 4.10's flowchart illustrates the basic workflow of a 3D gene expression clustering experiment in PCX. Initially, the user commonly explores the data using the visualization and manual data mining capabilities of PCX (see Section 4.2). Clustering is then used to investigate specific questions of interest. A 3D gene expression clustering experiment consists of several main steps. First, the user needs to define in the data selection step of the pipeline the data portions relevant for the clustering experiment (see Section 4.5.1.1). Afterwards, the user needs to specify mandatory input parameters for the clustering, such as, the number of clusters k. Finally, the actual clustering can be executed.

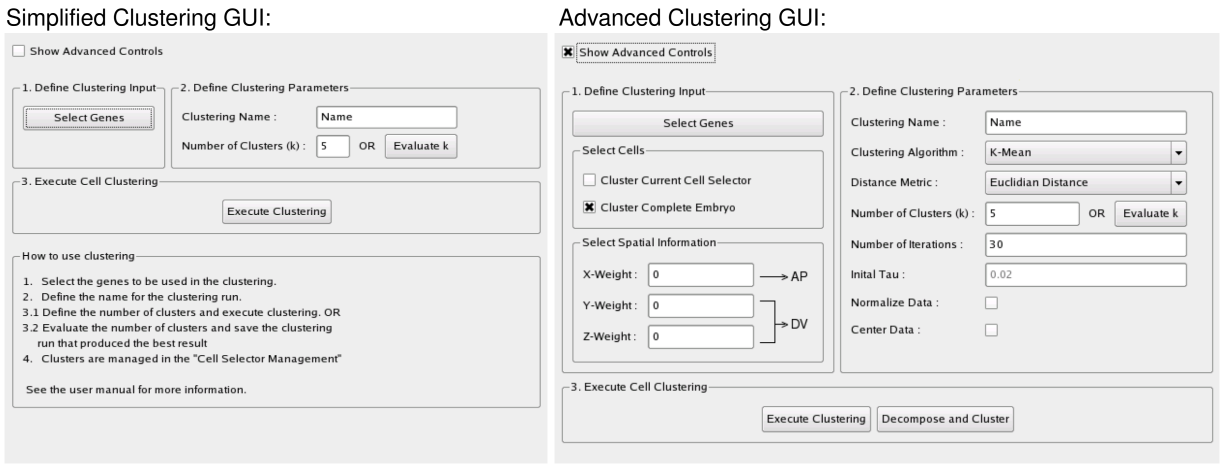
The clusters produced by a clustering experiment are then stored and managed by the central cluster management (see Section 4.5.2). The cluster management is an extension to the central cell selector management providing the user with a common interface for managing cell selections (whether automatically created clusters or manually created cell selectors). Besides general managing options, the user here also has access to advanced features such as cluster post-processing and cluster statistics. Cluster post-processing refers to a set of methods for correction and refinement of clustering results such as: i) merging and splitting of clusters, ii) manual correction and filtering of clusters, and iii) automatic re-coloring of clusters (see Section 4.5.2.1). Cluster statistics on the other hand are used for validation and analysis of clustering results (see Section 4.5.2.2).

Having completed a clustering experiment the user usually investigates and validates the clustering results using the visualization capabilites of PCX (see Section 4.5.3). Data clustering supports automatic highlighting of features of the data in the visualization enabling a much more detailed and focused analysis than a manual exploration procedure. Furthermore, data clustering enables the detection of fine feature not easily detected by eye or manual data mining.

4.5.1 Data Clustering in PointCloudXplore

The data clustering GUI of PCX follows the structure of the workflow of 3D gene expression clustering experiments as described above (see Figure 4.10). The clustering GUI is divided into three main areas (see Figure 4.11). First, the controls for performing the mandatory data selection step are presented in a box with the title *"1. Define Clustering Input."* A user here needs to define the parts of the data (i.e., genes and cells) relevant for the current clustering experiment (see Section 4.5.1.1). Second, a user needs to specify different parameters of the clustering, e.g., name of the clustering experiment and the number of clusters k (see Figure 4.11, *"Define Clustering Parameters"*). Having defined all necessary clustering inputs, the clustering can be executed using the *"Execute Cell Clustering"* controls (see Section 4.5.1.2). If the user does not have *a priori* knowledge of a good value for k then PCX provides dedicated features to support the user in the process of finding an appropriate number of clusters (see Section 4.5.1.2).

The target users of PCX include also researchers with no or little experience with data clus-



Simplified Clustering GUI:

Clustering Name : Name
Number of Clusters (k) : 5

1. Select the genes to be used in the clustering.
2. Define the name for the clustering run.
3.1 Define the number of clusters and execute clustering. OR
3.2 Evaluate the number of clusters and save the clustering run that produced the best result
4. Clusters are managed in the "Cell Selector Management"

See the user manual for more information.

Advanced Clustering GUI:

X-Weight : 0
Y-Weight : 0
Z-Weight : 0
Clustering Name : Name
Clustering Algorithm : K-Mean
Distance Metric : Euclidian Distance
Number of Clusters (k) : 5
Number of Iterations : 30
Inital Tau : 0.02

Figure 4.11: PointCloudXplore's GUI for data clustering supports two different modes to enable expert as well as non-expert users to make effective use of data clustering in their work. The simplified controls (left) ask the user only to specify the most important parameters. A "help text" is shown directly below the controls to guiding the user step-by-step through the clustering process. To give advanced users full control of the clustering process one can switch to the advanced controls simply by selecting the *"Show Advanced Controls"* ceckbox (top left). In the advanced mode (right) the simplified controls are extended, making all clustering parameters accessible to the user.

tering. PCX, therefore, provides a simplified clustering GUI in which the user is asked to specify only the most essential input values while other parameters are set to default values (see Figure 4.11 left). An expert user can extend these basic controls to access advanced clustering parameters and gain full control of the clustering process (see Figure 4.11 right).

4.5.1.1 Data Selection

While it is possible to execute the clustering algorithms on an entire dataset, a more typical use pattern is to focus clustering on a data subset relevant to a specific line of scientific inquiry. The researcher, therefore, needs to define which parts of the data are relevant to address the current problem. In order to define which parts of the expression data matrix are relevant, one needs to define: i) which rows (cells), and ii) which columns (gene+time point, x, y, z) are of interest. This section describes the different steps involved in the data selection process, as well as the effects of data selection on the cluster analysis, and describe how spatial information can be incorporated in the data analysis process. In the GUI the controls for data selection are located within the box *"1. Define Clustering Input"* (see Figure 4.11).

Cells of interest can be defined in PCX by using any cell selector or by using the results of

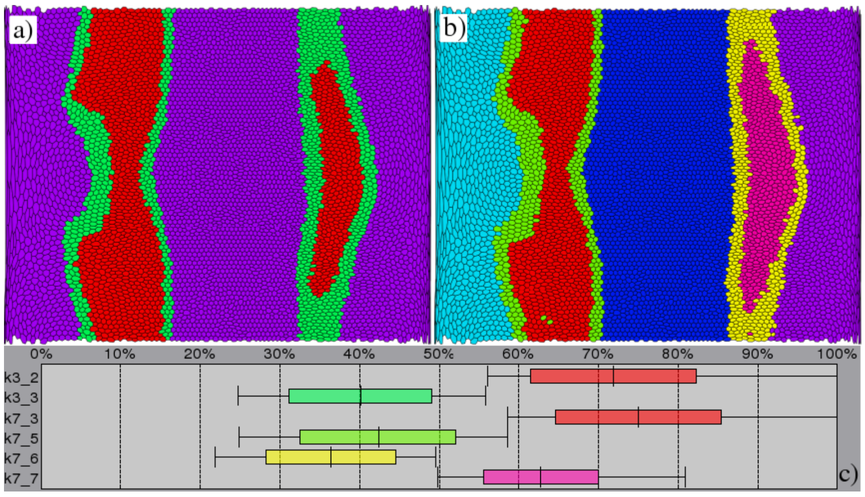


Figure 4.12: a) *giant (gt)* expression pattern classified using k-means clustering with Euclidean distances and $k = 3$. b) Same, using $k = 7$ and including x cell positions weighted with 0.24 (after normalization). c) A box-plot showing the statistics in *gt* expression (x-axis) for the two main clusters of the result shown in a) (first two entries on the y axis) and for the four main clusters of the clustering shown in b). Including spatial information in the clustering resulted in spatially separated clusters for the main regions of *gt*, as well as in different threshold levels depending on the physical cluster locations.

a previous data clustering as input to the clustering process. By default PCX assumes that all cells of the embryo should be clustered. The according option for defining the cells of interest is hence only shown in the advanced clustering controls (see Figure 4.11 right). Defining cells of interest focuses the analysis on a specific part of the data and also reduces the impact of surrounding noise on the analysis. By explicitly allowing data selection based on cell location, PCX overcomes one of the limitations of clustering methods designed for expression microarray data. By using an earlier clustering to define cell subsets of interest, one can first use PCX to group cells into a smaller number of clusters representing the predominant data features and then refine these clusters again using additional rounds of data clustering. In PCX, data clustering, as well as validation of clustering results, can in this way be performed in a step-by-step iterative process.

Defining which cell attributes are of interest is mandatory prior to clustering in PCX since these attributes define the actual biological context of the cells. To account for the complexities of 3D expression data, a variety of unique cell attribute data selection strategies is supported within PCX. First, genes of interest are generally identified based on visualization of the 3D gene expression data as well as based on input from other biological experiments, such as *in vivo* protein-DNA binding affinity data. Second, to account for spatial location in the clustering analysis, it is possible to directly use cell coordinates as input to the PCX clustering process. Adding this data enforces creation of spatially separated clusters along the AP (x) and/or the DV (y and z) body axes. Individual weights can be defined for x, y, and z. These weights are considered in the distance metric (see Section 4.4.1). However, in most cases the preferred way to incorporate spatial information in the analysis process is by splitting the newly computed clusters into their main independent spatial components (see Section 4.5.2.1). The main advantage of such a cluster post-processing technique over including cell coordinates in the clustering process is that cells with similar expression behavior in different parts of the embryo can be identified, and possible clustering artifacts due to the mixing of expression and spatial information can be prevented.

An improved quality of analysis results can be observed by adding spatial information to the clustering process when classifying the static pattern of a single gene that has a wide spatial distribution. Figure 4.12 shows as an example the classification of the pattern of the gene *giant* (*gt*) using k-means clustering with and without using x (AP) cell positions in the clustering process. In the first case, three clusters were created, each selecting cells expressing *gt* at different levels, i.e., low, medium, and high expression (Figure 4.12a). By considering x cell positions, the clustering was able to create separate clusters for the different major spatial components of the *gt* expression pattern (see Figure 4.12b). In this case, each cluster includes only cells that express *gt* at specific levels, while the minimum and maximum expression level selected by each cluster also depends on its physical location. In this case, higher threshold levels were created in the anterior, and lower thresholds in the posterior region of the embryo (see Figure 4.12c). Creation of region-dependent threshold levels is often desirable when analyzing the static pattern of a single gene since each domain of a pattern may be regulated differently and, therefore, different thresholds may be appropriate. For gap genes with spatially distant, independent expression domains, such as *gt*, this simple strategy works well, whereas for patterns with shorter interdomain distances, such as *eve*, this strategy fails.

4.5.1.2 Clustering

After having completed the data selection process the user is asked to specify various clustering parameters (see Figure 4.11, *2. Define Clustering Parameters*). In the case of the simplified clustering GUI the user only needs to specify a name for the clustering experiment and the number of clusters k. An expert user can then define additional clustering parameters in the advanced clustering controls, including the clustering algorithm and distance metric to be used, the number of iterations, and whether the data should be normalized and/or centered.

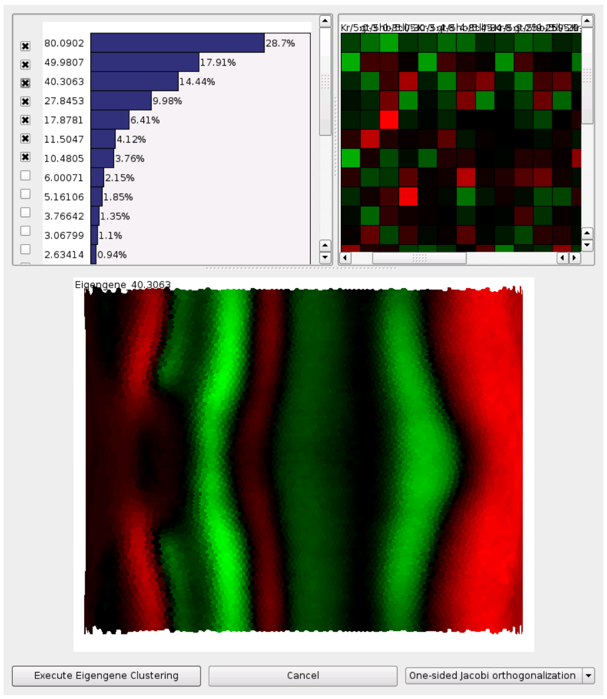


Figure 4.13: SVD GUI used for dimensions reduction. The percentage of original information represented by the according SVD eigen-genes are shown as bar-graph (top left). The relationship between SVD eigen-genes and the original patterns are indicated via a matrix-plot (tog right). Green indicates positive and red negative weights. The expression pattern of a user-selected eigen-gene is shown at the bottom of the window in an Unrolled View.

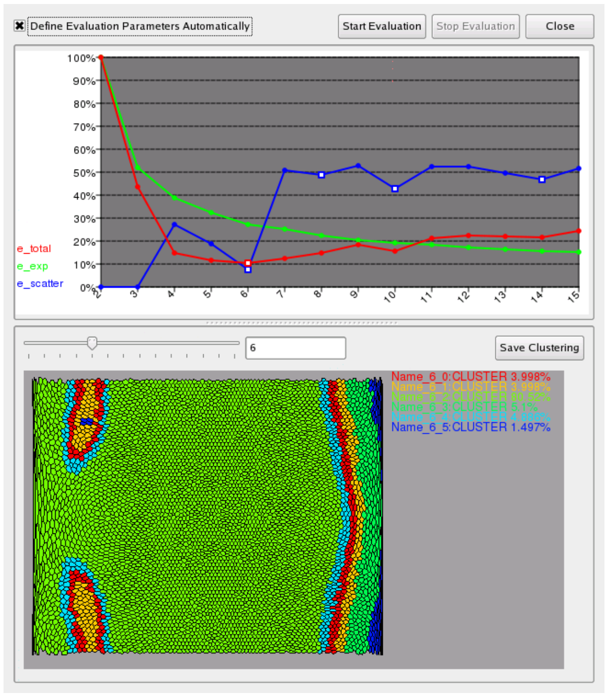


Figure 4.14: GUI for the evaluation of the number of clusters k. At the top, a plot of the cluster evaluation functions $\varepsilon_{\text{scatter}}$, ε_{exp}, and $\varepsilon_{\text{total}}$ is shown. After the evaluation is completed the suggested values are marked in the plot and a preview of the suggested clustering is shown at the bottom in an Unrolled View. Using a slider the user can and scroll through the evaluated clusterings by changing k and save any clustering of interest to cluster management.

To execute the actual clustering the user then may chose different paths. First, if the user has *a priori* knowledge of the number of clusters k, then the clustering can simply be executed directly (see Figure 4.11, *3. Execute Cell Clustering*). Second, if the user has *a priori* knowledge of the number of clusters k but the number of input genes is very large, then the advanced user may chose to perform dimension reduction prior to clustering (see Figure 4.11, *Decompose and Cluster*). In this case, singular value decomposition (SVD) [140, 141] is used to decompose the expression data matrix and only the user selected SVD eigen-genes are considered in the clustering process [160] (see Figure 4.13). Third, if the user has no *a priori* knowledge of the number of clusters k then PCX can assist the user in finding a good value for k (see Figure 4.11, *Evaluate k*).

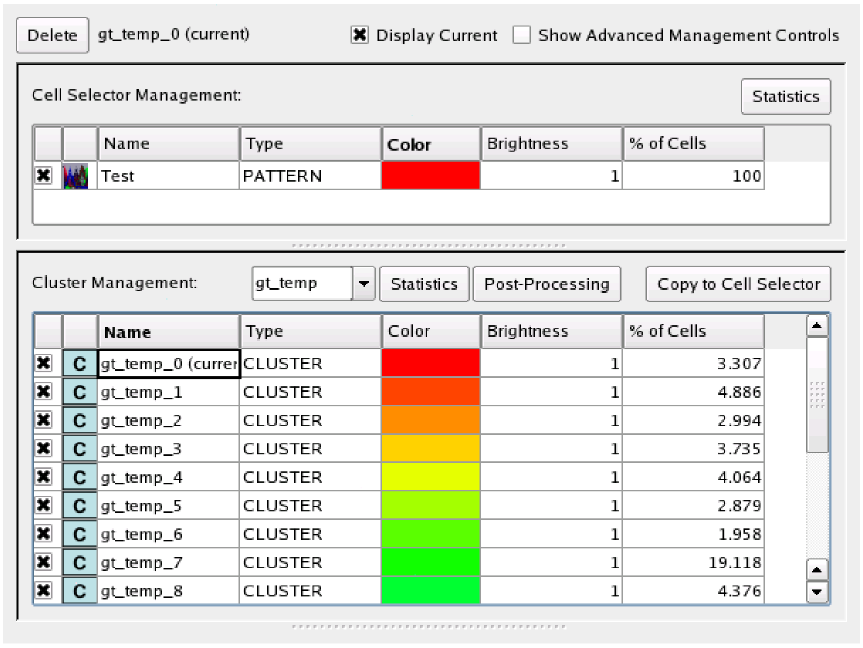
In the latter case, PCX will compute a series of clustering results with increasing k and evaluate the cluster quality functions $\varepsilon_{\text{scatter}}$, ε_{exp}, and $\varepsilon_{\text{total}}$ as described in Section 4.4.2. In this process the cluster evaluation plot of the three functions is constantly updated (see Figure 4.14 top) and an Unrolled View with a preview of the latest clustering is displayed (see Figure 4.14 bottom). The evaluation process can be terminated by the user at any time. If the user does not stop the evaluation manually then it will terminate automatically as soon as the predefined stop criteria are reached (e.g. $\varepsilon_{\text{scatter}} > 0.5$; see Section 4.4.2). Afterwards, all recommended values for k are marked in the cluster evaluation plot. The user can then scroll through the Unrolled View previews of the different evaluated clusterings and save any clustering result of interest.

4.5.2 Cluster Management and Analysis

Once created, clusters are stored and managed by a central cluster management system in PCX (see Figure 4.10). Clusters are implemented in PCX as a special type of POSITION cell selectors selecting cells directly via their unique identifiers (IDs). Clusters can, hence, be managed, visualized, and edited in PCX in the same way as POSITION cell selectors. In contrast to manually created cell selectors, automatically created clusters are, however, associated with a specific clustering experiment. Information about which clustering experiment resulted in the creation of a cluster as well as the used clustering settings (e.g., the genes used in the clustering, etc.) are essential to ensure reproducibility of the analysis and allows for effective grouping of clusters. Furthermore, clustering results need to be validated by, e.g., investigating the compactness of clusters.

To account for these special requirements, PCX provides a dedicated cluster management system (see Figure 4.15). The cluster management is a direct extension of the central cell selector management providing the user with a central interface for managing all types of cell selections. Consistent with the cell selector management, a table view is used for management of clusters (see Figure 4.15). In the table view a user can edit general properties of cluster, such as, cluster color, and review the settings used to create a cluster. To ease management of a large number of clusters, a user can chose to display either all, or only those clusters created during a specific clustering experiment in the table view.

Besides general management option, the cluster management also provides access to advanced features such as cluster post-processing (see Section 4.5.2.1) and cluster statistics (see Section 4.5.2.2). Cluster post-processing refers to a set of methods used for improvement and refinement of clustering results whereas cluster statistics provide means for validation and analysis of clusters.


Delete gt_temp_0 (current) ☒ Display Current ☐ Show Advanced Management Controls

Cell Selector Management: Statistics

		Name	Type	Color	Brightness	% of Cells
☒		Test	PATTERN	🟥	1	100

Cluster Management: gt_temp Statistics Post-Processing Copy to Cell Selector

		Name	Type	Color	Brightness	% of Cells
☒	C	gt_temp_0 (currer	CLUSTER	🟥	1	3.307
☒	C	gt_temp_1	CLUSTER	🟥	1	4.886
☒	C	gt_temp_2	CLUSTER	🟧	1	2.994
☒	C	gt_temp_3	CLUSTER	🟨	1	3.735
☒	C	gt_temp_4	CLUSTER	🟨	1	4.064
☒	C	gt_temp_5	CLUSTER	🟩	1	2.879
☒	C	gt_temp_6	CLUSTER	🟩	1	1.958
☒	C	gt_temp_7	CLUSTER	🟩	1	19.118
☒	C	gt_temp_8	CLUSTER	🟩	1	4.376

Figure 4.15: In PCX, the cluster management is integrated directly with the GUI of the central cell selector management. Common controls, e.g., for deleting a selected cell selector or cluster, are displayed at the top. Below these controls, the cell selector and cluster management are shown in two separate boxes. Besides general management options to define, e.g., the color of a cluster, the user here also has access to dedicated features such as cluster statistics and post-processing.

4.5.2.1 Cluster Post-processing

Cluster post-processing is essential to allow users to modify clustering results with respect to validation results or prior knowledge. There are four ways to post-process clusters in PCX. Manual correction and cluster filtering are two ways to correct small groups of misclassified cells. Cluster merging and splitting provide means to derive coarser or finer representations based on spatial information from the initial clustering.

Manual correction of clustering results can be performed in any physical view. By drawing on the embryo surface, one can interactively add and erase cells from the selection defined by a cluster. In contrast, filtering provides an automatic way to correct misclassified cells. Because genes are usually expressed in coherent spatial patterns, outliers in physical space tend to be also outliers in gene expression space.

Cluster filtering is an automatic method that identifies and reassigns misclassified cells to the spatially neighboring cluster that is closest in expression space. First, the filtering identifies all spatially independent components of a cluster that consist of less than *M* cells. To rule out false

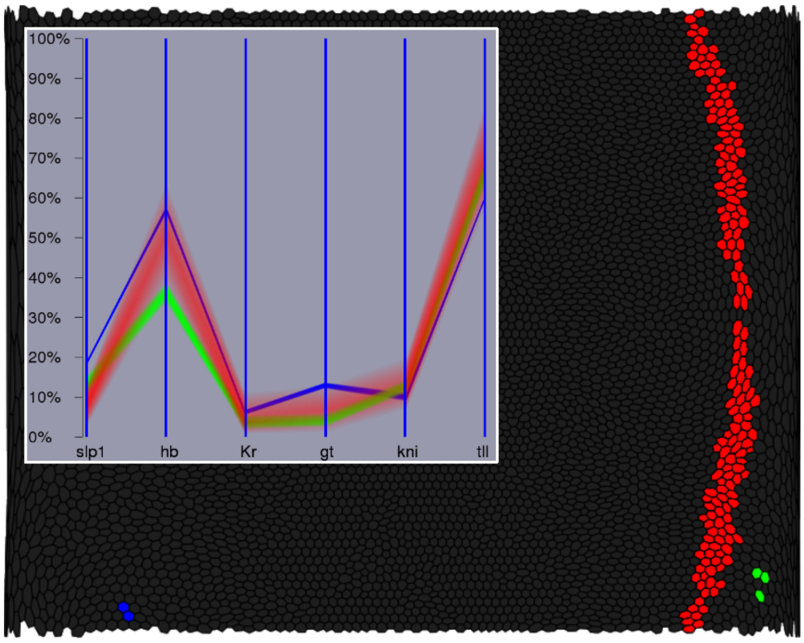


Figure 4.16: Filtering applied to an example cluster. The cluster is split into its four independent spatial components (red, blue, and two shown in green). The profiles of these regions in gene expression space are shown in parallel coordinates. The genes *slp1*, *hb*, *Kr*, *gt*, *kni*, and *tll*, which were used to obtain this clustering result, are each represented by one axis in the plot, and the percentage of expression is shown in ordinate direction (y-axis). One can see that the blue cells are spatially more distant to the main component of the cluster (red) than the green cells, and that they show a higher divergence from the main spatial component of the cluster in gene expression space.

filtering, a minimum distance in physical space as well as a maximum error in expression space can be defined. In the example shown in Figure 4.16, it would be possible to exclude the cells shown in green from the filtering process either by increasing the minimum spatial distance or by reducing the maximum allowed error in expression space. Having identified the misclassified cells the filtering then reassigns these cells to the spatially neighboring cluster that is closest in expression space

Cluster merging allows two clusters to be combined to form a single cluster. Merging clusters allows coarser representations to be created from an initial finer clustering. Such coarser descriptions often provide a clearer visualization that focuses on the main question being addressed (see, e.g., Section 4.7.2).

Cluster splitting, on the other hand, provides means to refine a cluster by dividing it into a set of clusters defining its main spatial components. Splitting clusters hence provides means to derive finer representation from clusters based on spatial information. A cluster often consists of several spatially independent components (for example Figure 4.24), which may need to be treated differently in subsequent analysis. In general, however, one major component of a cluster may be defined by a number of small spatially independent components. PCX uses a modified single linkage clustering approach to split up such a cluster into a selected (often smaller) number of components.

The splitting algorithm works as follows: A cluster is first split into all its spatially independent components. The smallest components are subsequently merged with the spatially closest component. This approach is computationally more efficient and less sensitive to outliers than a classical single linkage clustering and also guarantees that the independent spatial components of a cluster are preserved while small, scattered components can still grow to define major cluster components. An example for cluster splitting is described later in Section 4.7.2.

4.5.2.2 Cluster Statistics and Validation

Analysis of statistical properties of clusters is essential for both the validation and analysis of clustering results. Cluster properties provided by PCX include the percentage of cells selected by a cluster, as well as the minimum, maximum, average, and standard deviation values for gene expression levels in a cluster. In addition, histograms provide the user with information about the distribution of expression values within clusters. A valid clustering is expected to produce compact clusters in the the gene dimensions used for clustering. Using cluster statistics a user can effectively investigate the compactness of clusters in expression space and validate clustering results. Besides for validation, cluster statistics play also an important role in the analysis of clustering results. Average curve plots, e.g, are often used to investigate the temporal variation of genes within clusters.

To compare these statistical properties for one gene in multiple clusters or multiple genes in one cluster, PCX provides box-plots and multi-dimensional color/transparency histogram plots. The histogram plots use both color and transparency to visualize the number of cells within a cluster that express the gene over a range of expression levels. Average curve plots (with optional error bars showing standard deviation values) aid in simultaneous analysis of multiple clusters in multiple genes. For convenience a 2D scatter plot and an Unrolled View are also made available in the cluster statistics window enabling a more efficient workflow.

Figure 4.17 shows a simple example in which various cluster statistic plots are used to investigate expression characteristics of different parts of the *giant (gt)* expression pattern. More examples illustrating the use of cluster statistic in the context of realistic use case examples are provided later in Section 4.7.3 and 4.7.4.

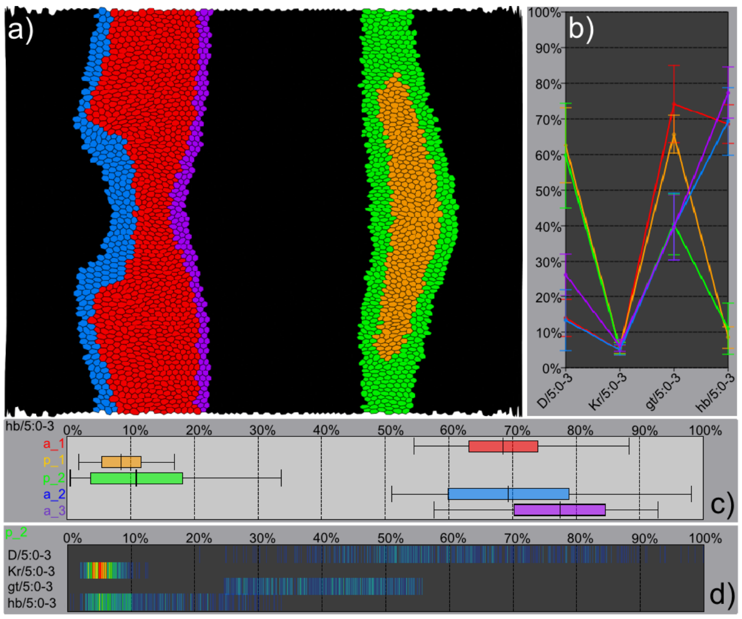


Figure 4.17: An analysis of characteristics of the *giant (gt)* expression pattern using cluster statistics. a) An unrolled view showing the spatial structure defined by five clusters. The red and orange cluster define the centers of the two expression regions of *gt*, and the other clusters define the boundaries. b) A curve plot showing the average expression profiles of the genes *D*, *Kr*, *gt*, and *hb* in each of the five clusters (x-axis). The y-axis represents expression level. c) A Box-plot comparing the expression of *hb* in the five clusters. The x-axis represents expression level. d) A color/transparency histogram comparing the expression of *D*, *Kr*, *gt*, and *hb* for cluster p_2 (green). The x-axis indicates gene expression level. A heat-map coloring scheme is used to indicate the number of cells in the cluster having a given expression level: red indicates many cells, while blue indicates few cells.

4.5.3 Cluster Visualization

As described earlier, clusters are implemented in PCX as a special type of cell selector. Automatically created clusters can, hence, be visualized in PCX in the same way as regular cell selectors. The most common way to visualize clusters (and cell selectors) in PCX is through the use of a consistent color mapping. In the visualization, PCX allows colors to be assigned to clusters either randomly, manually, according to size, or according to the average or ranked average expression of selected gene(s) in the clusters.

Using physical views, the spatial pattern defined by a cluster can be analyzed. By mapping cluster colors onto the expression surface of a gene, a user can effectively compare the spatial patterns defined by a set of clusters with the spatial expression pattern of a gene. This approach provides a user with a qualitative notion of the goodness of fit of patterns predicted by the clustering and the spatial patterns defined by the expression of suspected target genes (see, e.g., Section 4.7.4 and 4.7.6).

Abstract views allow for identification of cluster characteristics in gene expression space. Scatter plots provide the user with an intuitive visualization revealing the location and structure of clusters in 2D/3D gene expression space. Parallel coordinates are then used to analyze cluster characteristics in n-dimensional gene expression space. PCX supports cell-selector bands and other dedicated extensions to standard parallel coordinates enabling the comparison of expression characteristics of multiple clusters in multiple gene expressions (see Section 4.2.2.3).

4.6 Integrating MATLAB and PointCloudXplore for the Analysis of 3D Gene Expression Data

MATLAB provides a rich set of visualization and analysis tools. However, this functionality is not readily accessible to the average biologist with little or no programming experience. Furthermore, MATLAB is designed as a general purpose analysis environment and not specifically for the analysis of 3D gene expression data. To overcome these limitations, this work extends PointCloudXplore (PCX) so that it can be used as a direct interface to MATLAB with the goal to: i) make advanced analysis functions defined in MATLAB more readily accessible to PCX users, ii) enable MATLAB developers to directly integrate their analysis with the visualization to ease validation and deployment of their code, iii) enable researchers to flexibly extend PCX, and iv) to provide an effective link between bioinformatics researchers and their target users. Ultimately this means that: i) PCX users should not be required to know MATLAB in order to use analysis functions defined in MATLAB, and ii) developers should not be required to modify PCX (or know its internal design) in order to integrate their analysis functions defined in MATLAB with PCX.

To achieve these goals, PCX implements an interface for direct communication with MATLAB based on MATLAB's C API. Using this interface, PCX can start and close a MATLAB environment, import/export data from and to MATLAB, and initiate MATLAB function calls. This interface allows PCX to perform all communications necessary to support MATLAB-based analysis.

In order to make a MATLAB function accessible to PCX, a developer needs to create: i) a *M* file with the function specification[4], and ii) a so called *PCXM* file with a description of the

[4]*M* files are the standard format of MATLAB to define functions that can accept input arguments and return output arguments. Functions specified in *M* files operate on variables within their own workspace. The name of the

function (see Appendix A for details). The *M* file defines the implementation of the actual analysis function whereas the *PCXM* file describes how PCX should represent and call the function. In the *PCXM* file the developer needs to specify: i) a name (*NAME*) and type (*TYPE*) for the function, ii) the path (*DIR*) to the *M* file, and iii) the MATLAB command (*CALL*) to be used to call the function (see Appendix A.2 for details). The name and type of a function are used to represent the function in the GUI of PCX whereas the *DIR* and *CALL* parameter are used to call the function.

Based on the *CALL* string PCX automatically identifies which inputs a function expects and which outputs it will create. In order to do this, PCX specifies a set of name conventions (see Appendix A.3) for the in- and output parameters. PCX supports a wide range of in- and output options (see Appendix A.3) and can practically send all the data of a PointCloud to MATLAB, e.g., i) gene expressions and gene names, ii) cell positions, iii) projected cell positions form the Unrolled View, iv) cell neighbors as defined in the Cell View, or v) the filename. In addition PCX can also send to MATLAB cell selectors and their names as well as any double, integer, or Boolean input parameters specified as user input. The data PCX sends to MATLAB can in turn be directly accessed by the according MATLAB function. An analysis function then may produce the following outputs to be imported by PCX: i) new derived-gene expressions, ii) optional names for new derived-gene expressions, iii) new cell selections to be stored as cell selectors, and iv) optional names for new cell selectors. All in- and output parameters are optional and in general any combination of in- and output parameters is valid. A detailed description of PCX's interface to MATLAB is provided in Appendix A.

PCX's MATLAB interface allows the integration of MATLAB functions with PCX without the need to re-compile any code, i.e., PCX does not require any information about the MATLAB functions it should access during compile-time. On start-up PCX parses all *PCXM* files located in the current function path. Based on the information of the *PCXM* files, PCX automatically creates a menu bar listing the accessible MATLAB functions grouped by their type. From this menu bar a user can conveniently access all available MATLAB functions. Upon calling, PCX automatically creates at runtime a dedicated GUI allowing the user to specify all necessary input parameters of the according analysis function. During execution of the analysis, PCX automatically handles all necessary communication with MATLAB (e.g., send input data to MATLAB, call the function, and import data from MATLAB).

The fact that all operations necessary to integrate a MATLAB function into PCX are performed at runtime allows for great flexibility with respect to the management of external analysis

M-file and the function its specifies should be the same.

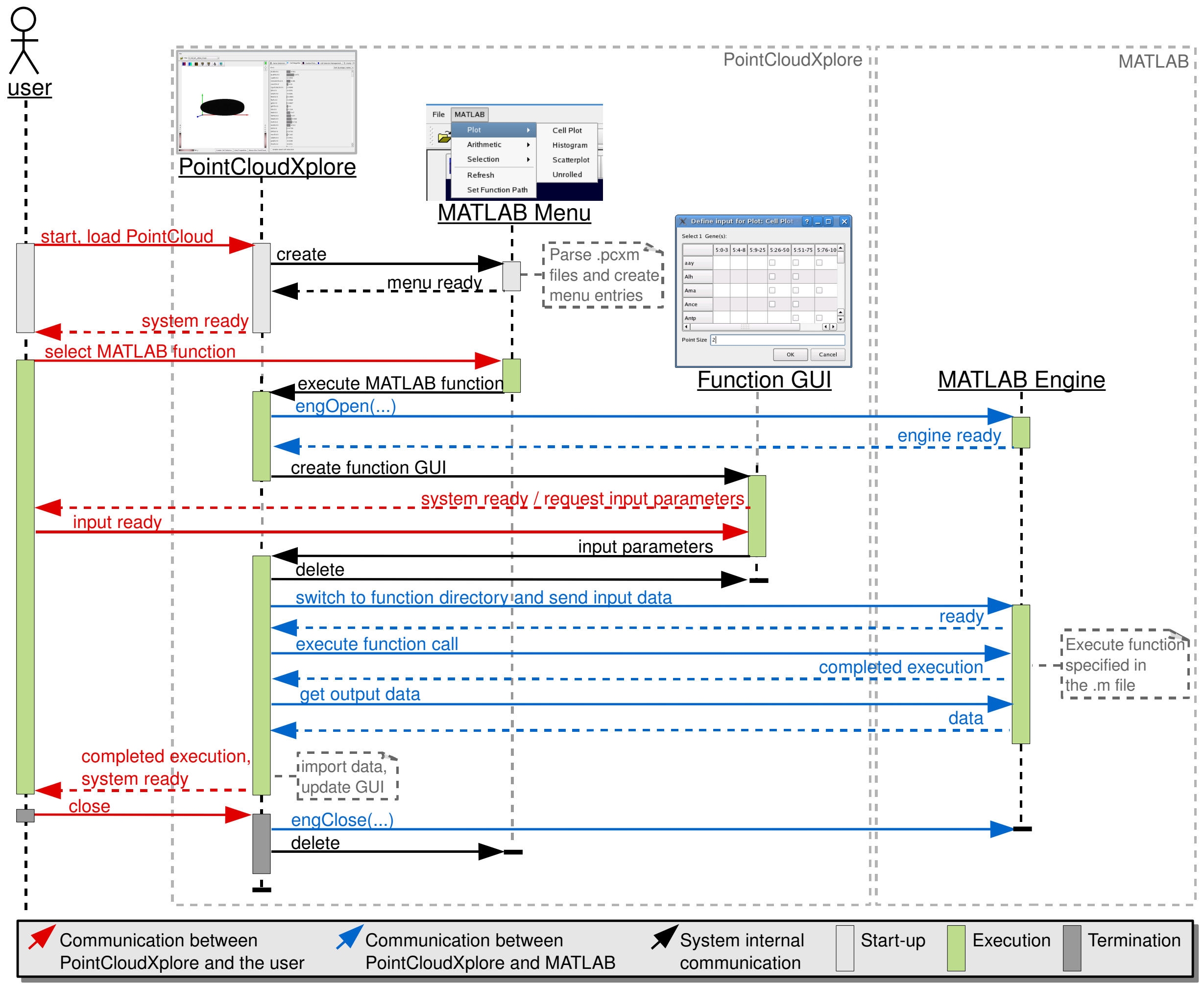


Figure 4.18: Sequence diagram illustrating the workflow of a typical use case scenario involving PointCloudXplore (PCX) and MATLAB. The sequence is divided into three main steps: i) start-up, ii) execution, and iii) termination. Note that the user only interacts with PCX itself. All communications with MATLAB are handled automatically by PCX.

functions. A user, e.g., may change the path to the folder were the external analysis functions are located at any time, i.e., the user can switch between different sets of external analysis functions located in different folders at runtime. A developer can also, e.g., update an analysis function at runtime and test it immediately without having to restart PCX. In fact, as long as only the implementation (i.e., the *M* file) but not the description (i.e., the *PCXM* file) of a function changes, PCX does not need to be updated at all since the function itself is dynamically accessed by MATLAB upon execution.

Figure 4.18 illustrates the structure of the PCX/MATLAB-interface and the general workflow of a typical use case scenario involving both, PCX and MATLAB. On start-up, PCX first parses all *.pcxm* files located at the current function path and dynamically creates the *MATLAB menu* bar listing all available externale analysis functions. When the user requests the execution of a MATLAB analysis function PCX first starts a MATLAB engine if it is not already running. Afterwards, PCX creates at runtime a dedicated GUI according to the description of the analysis function allowing the user to specify all required input values directly from within PCX. PCX then sends the necessary data to MATLAB and calls the analysis function. After completion of the analysis, PCX imports the analysis results, such as new derived-gene expressions or cell selections. The user can then investigate the analysis results using PCX. In case that MATLAB is used for visualization then the according plot will be displayed in a separate window created and managed by MATLAB itself. When PCX is terminated it will automatically also close the MATLAB engine it started.

In the whole analysis process no direct interaction between MATLAB and the user is required. The user only interacts directly with PCX which in turn automatically handles all required communication with MATLAB. PCX give the user in this way access to advanced MATLAB analysis functions while effectively hiding the complexity of using MATLAB from the user, i.e., from a users perspective MATLAB appears as a black box.

From a developer's perspective, PCX's interface to MATLAB allows an effortless integration of analysis functions defined in MATLAB with PCX. Developers usually already package their analysis functions in *M* files so that the developer only needs to ensure that the interface of the function fits the requirements (i.e., the name conventions) of the PCX/MATLAB interface. In addition the developer only needs to create an according *PCXM* file describing the function for PCX. Most importantly, a developer does not need to (re)compile any code in order to integrate a MATLAB function into PCX. As illustrated in the examples described in Appendix A, the additional work required to integrate a new MATLAB function into PCX is usually minimal.

The interface between PCX and MATLAB has a wide range of possible applications, e.g., i) plotting, i.e., use MATLAB to create dedicated data visualizations (see Appendix A.4.1), ii) selection, i.e., use MATLAB for automatic cell selection and feature detection (e.g., via clustering) (see Appendix A.4.3), or iii) filtering, i.e., use MATLAB to define new derived-gene expressions. Expression filtering itself has a wide range of applications. Besides smoothing of expression patterns or noise reduction, expression filtering can, e.g., be used for modeling of genetic regulatory networks (see Sections 4.7.5), analysis of the temporal variation of genes (see Section 4.7.6), or validation and improvement of clustering results (see Section 4.7.6).

4.7 Results and Applications

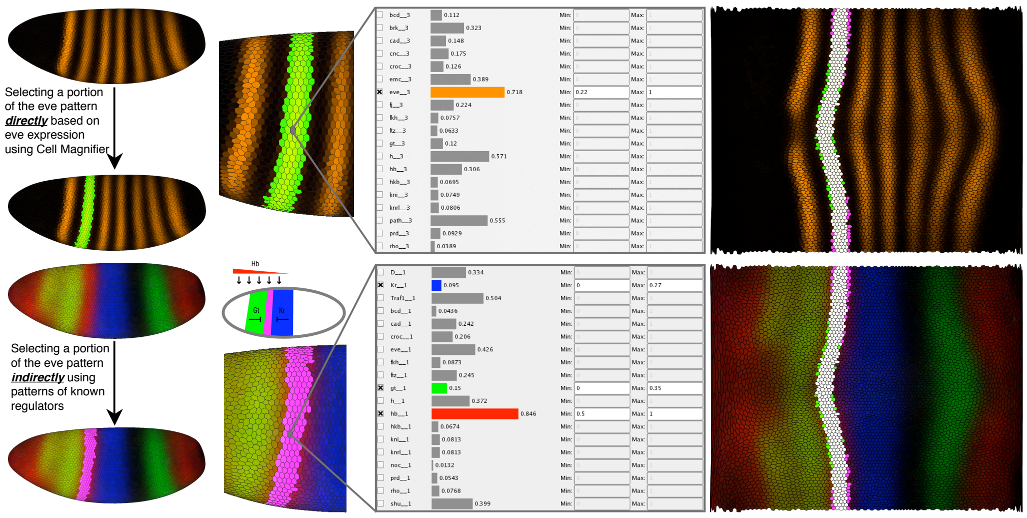
This section discusses several examples illustrating the use of the proposed framework for knowledge discovery from 3D gene expression data. Section 4.7.1 describes the exploration of regulatory relationships between genes using manual data mining based on visualization. Afterwards several examples are presented illustrating the effectiveness of data clustering and visualization to address the following questions: i) How can cells usefully be divided into distinct components of a gene's expression pattern? (see Section 4.7.2); ii) What is the temporal variation of a gene expression pattern? (see Section 4.7.3); and iii) What components of a gene's expression pattern are related to the expression patterns of the regulatory factors that control it? (Section 4.7.4). Finally, the last two examples describe the use of MATLAB and PointCloudXplore (PCX) for modeling of genetic regulatory networks (see Sections 4.7.5) and analysis of the temporal variation of a gene expression pattern (see Section 4.7.5).

4.7.1 Exploring Regulatory Relationships Using Manual Selection

The examples described in this section illustrate the exploration of regulatory relationships between genes using the advanced visualization and manual data mining features of PointCloudXplore (PCX). Cell selection in physical views limits the data set into specific parts of the embryo. The combination with abstract views, like scatter plots or the Cell Magnifier, help in more rigorous selection of studied expression values.

For example, the upper panel of Figure 4.19 shows the use of the Cell Magnifier and Seed Cell Selection to select a single *eve* stripe for further analyses. The bottom panel shows the inverse; by selecting three known regulators of *eve*, *gt*, *hb* and *Kr* (bottom left panel), one can more or less reproduce the selected *eve* stripe, as shown in the right panel. Importantly for biologists, the same method can be used for identifying novel interactions between genes with interesting spatial patterns or scatter plot correlations. Using PCX to first identify candidate genes for later experimental validation will often be much cheaper than the traditional methods for selecting candidate genes, such as mutagenesis screens or by staining for co-expression for each gene pair, although there are cases when the older methods are more useful.

Interestingly, the same results can be obtained in multiple ways, to suit the questions of the biologists using the tools. As Figure 4.20 shows, *eve* stripe 2 can be detected also using the whole embryo expression data with help of logical operators. However, when not limiting the data spatially, additional details are often revealed. In this case, the logical combination of *gt*, *hb* and *Kr* expression also follow the anterior border of *eve* stripe 7. While regulation of *eve* stripe


Selecting a portion of the eve pattern *directly* based on eve expression using Cell Magnifier
Selecting a portion of the eve pattern *indirectly* using patterns of known regulators
Hb
Gt
Kr

☐	bcd__3	0.112	Min: 0	Max: 1
☐	brk__3	0.323	Min: 0	Max: 1
☐	cad__3	0.148	Min: 0	Max: 1
☐	cnc__3	0.175	Min: 0	Max: 1
☐	croc__3	0.126	Min: 0	Max: 1
☐	emc__3	0.389	Min: 0	Max: 1
☒	eve__3	0.718	Min: 0.22	Max: 1
☐	fj__3	0.224	Min: 0	Max: 1
☐	fkh__3	0.0757	Min: 0	Max: 1
☐	ftz__3	0.0633	Min: 0	Max: 1
☐	gt__3	0.12	Min: 0	Max: 1
☐	h__3	0.571	Min: 0	Max: 1
☐	hb__3	0.306	Min: 0	Max: 1
☐	hkb__3	0.0695	Min: 0	Max: 1
☐	kni__3	0.0749	Min: 0	Max: 1
☐	knrl__3	0.0806	Min: 0	Max: 1
☐	path__3	0.555	Min: 0	Max: 1
☐	prd__3	0.0929	Min: 0	Max: 1
☐	rho__3	0.0389	Min: 0	Max: 1

☐	D__1	0.334	Min: 0	Max: 1
☒	Kr__1	0.095	Min: 0	Max: 0.27
☐	Traf1__1	0.504	Min: 0	Max: 1
☐	bcd__1	0.0436	Min: 0	Max: 1
☐	cad__1	0.242	Min: 0	Max: 1
☐	croc__1	0.206	Min: 0	Max: 1
☐	eve__1	0.426	Min: 0	Max: 1
☐	fkh__1	0.0873	Min: 0	Max: 1
☐	ftz__1	0.245	Min: 0	Max: 1
☒	gt__1	0.15	Min: 0	Max: 0.35
☐	h__1	0.372	Min: 0	Max: 1
☒	hb__1	0.846	Min: 0.5	Max: 1
☐	hkb__1	0.0674	Min: 0	Max: 1
☐	kni__1	0.0813	Min: 0	Max: 1
☐	knrl__1	0.0813	Min: 0	Max: 1
☐	noc__1	0.0132	Min: 0	Max: 1
☐	prd__1	0.0543	Min: 0	Max: 1
☐	rho__1	0.0768	Min: 0	Max: 1
☐	shu__1	0.399	Min: 0	Max: 1

Figure 4.19: Using Physical Views and Cell Magnifier to examine regulatory relationships for *eve* stripe 2.

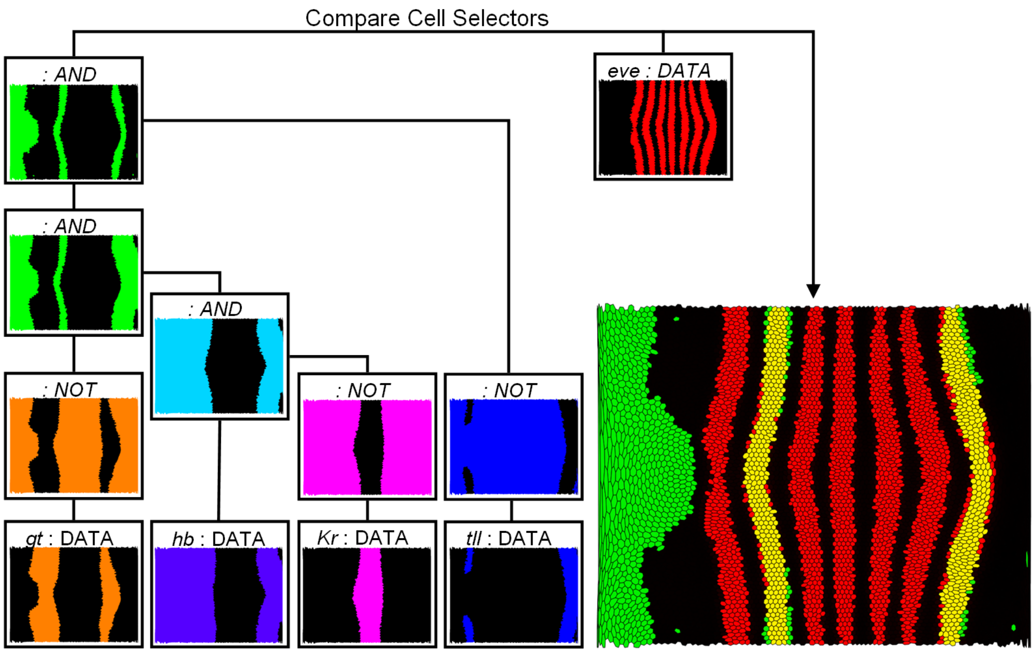


Figure 4.20: Using logical operations to examine regulatory relationships for *eve* stripe 2 and 7. The expression patterns of *giant (gt)*, *hunchback (hb)*, *Krüppel (Kr)*, and *tailless (tll)* are first classified by defining an independent cell selector in scatter plots (DATA cell selector) for each gene. Subsequently, the cell selectors defining the *gt*, *Kr*, and *tll* patterns are inverted using a NOT operation. Afterwards these logical cell selectors as well the cell selector defining the *hb* pattern are combined using a sequence of AND operations. In this way the overlap of the hb expression pattern, and the inverted *gt*, *Kr*, and *tll* expression patterns can be determined. The result (green) is compared to the *eve* expression pattern (red) identified by another DATA cell selector.

2 is often understood to be separate from *eve* stripe 7 regulation, actually there is some evidence that the stripe 7 regulatory region is partially connected to the stripe 2 regulatory region [142]. In this case, it makes sense that both stripes also correlate similarly with the expression of stripe 2 transcriptional regulators. Moreover, when adding *tll* expression to the equation, the posterior border of stripe 7 is obtained, in line with earlier 1D simulation results [142]. Notably, normal *tll* expression has been experimentally shown to upregulate the commonly cited minimal regulator of *eve* stripe 7, perhaps because of indirect effects, e.g., by inhibition of inhibitors of *eve* stripe 7 in this minimal regulator. This shows that though computational analysis of data can reveal interesting correlations that might be indicative of novel biological interactions, data mining results should be validated with real experiments. However, PCX is a useful data mining tool, not only for directing the expensive *in vivo* experimentation, but also for directing simulation experiments, where the expression analyses in PCX can additionally be used for validating the simulation results.

As theses examples show, an analysis based on manual exploration can be very revealing. However, manual data mining has its limitations. Using a single thresholds to describe expression patterns results in binary (on/off) pattern descriptions and, hence, loss of detail. For example, in a binary description of the expression pattern of *eve* information about the variation of the pattern along the DV circumference is lost (see, e.g., Figure 4.3). Furthermore, manual thresholding is arbitrary, i.e., thresholds are not fully data-dependent. The examples described in the following sections illustrate the use of data clustering and the PCX/MATLAB-interface to address these problems.

4.7.2 Single Pattern Analysis

Genes are frequently expressed in complex patterns that show a wide range of quantitative changes in expression across the cells of an embryo. Although for some analyses, the data is best left unclassified in this form —simply using the expression values in all cells— it can also be revealing to divide a single pattern into one or more distinct regions. For example, on/off descriptions of expression have been useful in logical models of gene networks [143, 144] (see also Section 4.7.1).

However, as described earlier, discretizing a gene pattern via manual thresholding can be problematic — it may be very time-consuming, and the choice of thresholds is arbitrary and not fully data-dependent. The specific problem of pattern segmentation is described in earlier work by Huang et al. [163] in which the authors compare different methods for creating binary (i.e., on/off) descriptions of expression patterns. To address the more general problem of discretizing

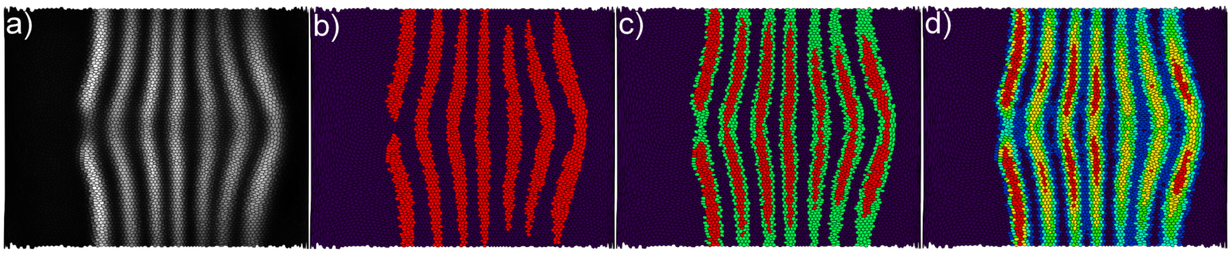


Figure 4.21: a) The expression pattern of *eve* at stage 5:9-25%; Classification of *eve* with b) $k = 2$; c) $k = 3$; and d) $k = 6$. While the $k = 2$ clustering produced a threshold that was too high, erasing too many cells from the pattern, the $k = 3$ clustering was better able to identify the seven stripes of the *eve* expression pattern. The $k = 6$ clustering identified additional characteristic variations within the stripes along the DV-axis as well as an additional cluster that selects some inter-stripe cells showing some higher expression of *eve*.

a gene pattern, one can use, for example, k-means clustering and Euclidean distances to compute a number of data-dependent thresholds. Each of the k clusters then represents a specific threshold range that can be interpreted as a different confidence level. Different components of a pattern may be regulated by different genes, so different thresholds may be appropriate for different regions. Cluster post-processing, such as splitting clusters into their main spatial components, allows different threshold levels to be selected for different components of a gene pattern. Alternatively, as described in Section 4.5.1.1, for genes with clearly distinct spatial expression domains, cell positions may be used in the initial clustering to enforce creation of separate clusters for spatially distant components of a pattern. Rather than choosing some arbitrary thresholds, clustering automatically suggests thresholds based on the histogram of the data. The k-means clustering algorithm seeks to minimize the mean squared distance from each data point (cell) to its nearest cluster center. To achieve this goal, the k-means algorithm will create k cluster centers positioned according to the density distribution of the expression values of the selected gene.

Figure 4.21 shows three example classifications of the *eve* expression pattern using different numbers of clusters k. While $k = 2$ produces a threshold that is too high and does not capture all parts of each stripe, a clustering with $k = 3$ correctly identifies the seven stripes of the *eve* expression pattern. By increasing the number of clusters, additional details within the stripes along the dorsal-ventral axis can be seen, as well as an additional cluster selecting cells in the inter-stripe regions. This complex description illustrates that thinking of a gene as being either on or off is usually too simplistic. The fact that clustering automatically reveals differences along the dorsal-ventral axis demonstrates the usefulness of such analyses. The pair-rule genes, such as *eve*, are not typically thought of as dorsal-ventral regulators, but consistent with the clustering results, careful quantitation of the levels of *eve* and similar gene's expression has shown they

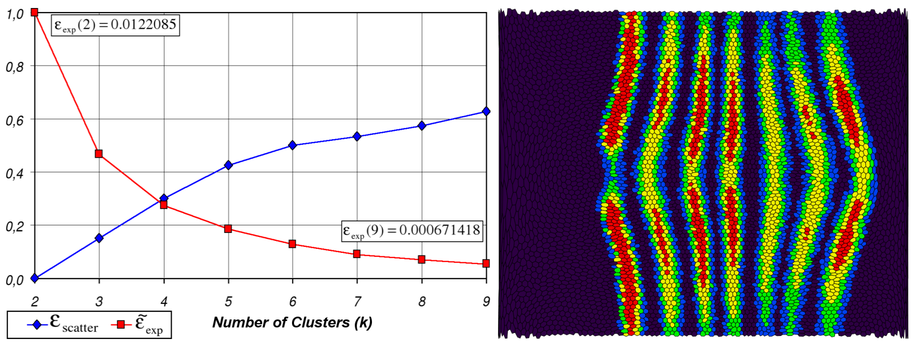


Figure 4.22: Cluster evaluation functions $\tilde{\varepsilon}_{\text{exp}}$ (red) and $\varepsilon_{\text{scatter}}$ (blue) for the clustering of the *eve* expression pattern (left), with $w = 4$ and $m = 9$. The suggested value for k is five, which is the largest value of k for which only one cluster representing low background *eve* expression is created (right).

indeed show up to two-fold changes in expression along the dorsal-ventral axis suggesting a dorsal-ventral component in pair-rule regulation [99, 94]. Analyzing the actual meaning of these moderate changes requires computational tools, such as cluster analysis, to provide objective measures of their significance.

Figure 4.22 (left) shows the curves of the cluster evaluation functions $\varepsilon_{\text{scatter}}$ and $\tilde{\varepsilon}_{\text{exp}}$. In this case, $\varepsilon_{\text{scatter}}$ is rather smooth and monotonically increasing indicating that all k with $\varepsilon_{\text{scatter}} > 0$ may result in valid clusterings of the *eve* pattern. This behavior can be explained by the very high signal-to-noise ratio of the *eve* expression data, which was averaged from dozens of embryos. The suggested value for k is five, which is the largest value of k for which only one cluster representing low *eve* expression is created (see Figure 4.22, right). A clustering with $k = 5$ provides a compromise between a high-level and low-level description of the *eve* expression pattern.

Binarized versions of the *eve* pattern (i.e., on/off descriptions) can be created by merging the different clusters, allowing one to easily compare the different classifications by defining their overlay (see Figure 4.23). While $k = 3$ and $k = 6$ result in similar classifications of the seven stripes, the clustering with $k = 2$ misses many cells of the pattern. Thus, first generating multiple clusters and then merging them can provide a more accurate binarization of an expression pattern than an initial $k = 2$.

Cluster merging and splitting can also be useful for comparing different gene patterns or for comparing different components of a single gene's pattern. In Figure 4.25, for example, the individual clusters shown in Figure 4.21d have been merged and then split to obtain one cluster representing each stripe. Figure 4.24 shows an example where the cluster that defined

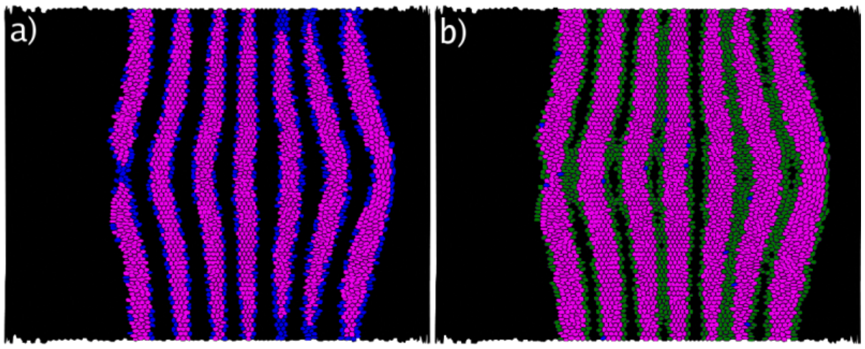


Figure 4.23: Comparison of the clustering results shown in Figure 4.21. a) Comparison of $k = 2$ (red) and $k = 3$ (blue) classification of *eve*; b) Comparison of $k = 3$ (red) and $k = 6$ (blue) classification of *eve*. In b), the additional inter-stripe cluster found in the $k = 6$ clustering is shown in dark green. The percentage of cells of the whole embryo selected by the different components are: $k2 = 31.31\%$; $k3 = 42.596\%$; $k6 = 42.892\%$; $k3 - k2 = 11.287\%$; $k6 - k3 = 0.296\%$; $k6\_interstripe\_cluster = 21.06\%$.

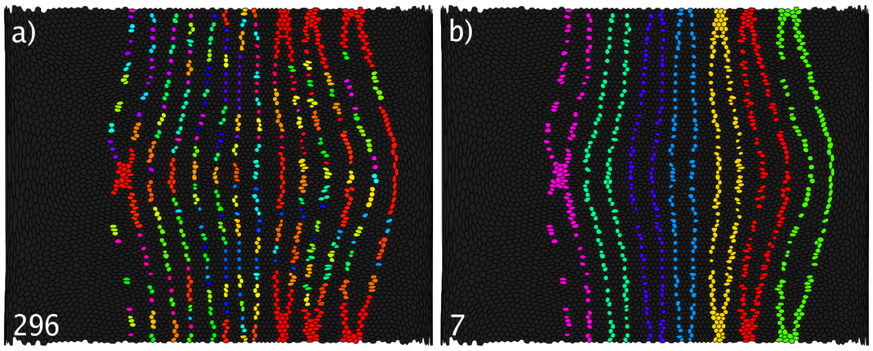


Figure 4.24: a) A cluster consisting of 296 spatially independent components; b) The same cluster split into its seven main spatial components. Splitting of clusters is essential, e.g., to allow comparison of different main spatial components of a cluster.

the boundary of the stripes, consisting of 296 spatially independent components, is split into its seven main components using the modified single linkage method described in Section 4.5.2.1.

Once derived, these individual stripe clusters are used to highlight the seven *eve* stripes via color in different abstract views. For example, the expression behavior of *gt*, *hb*, and *Kr* –three known transcriptional regulators of *eve*– can be revealingly analyzed within each of the *eve* stripes using a 3D scatter plot (Figure 4.25). In the plot large differences between stripes are visible, the seven stripes form very distinct point clusters within the scatter plot. This behavior is consistent with current models suggesting that the *eve* expression pattern does not simply consist of seven identical stripes, but that many stripes are regulated independently. The available data suggests that *gt*, *hb*, and *Kr* control some stripes, but the scatter plot suggests that these

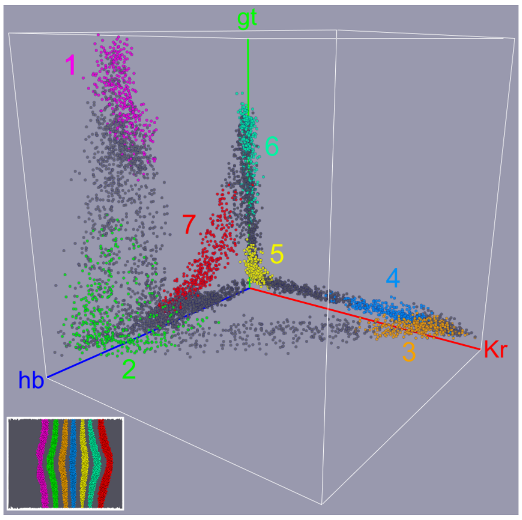


Figure 4.25: An Unrolled View showing seven clusters, each selecting one stripe of the *eve* expression pattern (bottom left). The same clusters shown in a scatter plot of early-stage *Kr* (red), *gt* (green), and *hb* (blue). Color indicates to which cluster a cell belongs, while cells not selected by any cluster are colored gray. The stripes form characteristic clusters in expression space indicating a potential relationship between *eve* and the displayed genes.

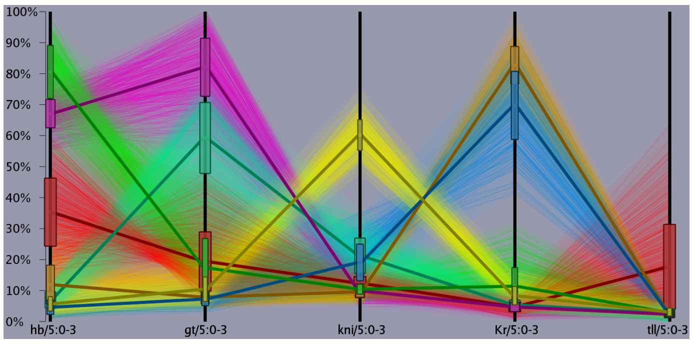


Figure 4.26: The same clusters as in Figure 4.25 are shown in a 2D parallel coordinate view of early stage *hb*, *gt*, *kni*, *Kr*, *tll*. The average expression of the seven clusters in the different genes are shown via additional, thicker lines of darker colors and the associated standard deviations are shown via boxes placed on each parallel axis. Highly transparent color bands shown in front of the plot are used to further highlight the different clusters.

factors have the potential to regulate all stripes by their unique combinations of expression levels. Such plots can be very useful in identifying potential novel regulatory relationships between transcription factors and their targets.

Generally, scatter plots have proven to be a very intuitive and informative gene expression space visualization, but are limited due to the fact that only three gene dimensions can be visualized at once. PointCloudXplore (PCX) also provides 2D and 3D parallel coordinates to support simultaneous visualization of many more genes [157]. In Figure 4.26, the same clusters as in Figure 4.25 are shown in a 2D parallel coordinate view of early stage *hb*, *gt*, *kni*, *Kr*, and *tll* indicating additional expression differences between the spatial clusters. Because numerical PointCloud datasets are not easily comprehensible, the clustering and cluster manipulation capabilities in PCX provide a reasonably objective method for dividing quantitative spatial expression data into units that can be analyzed computationally.

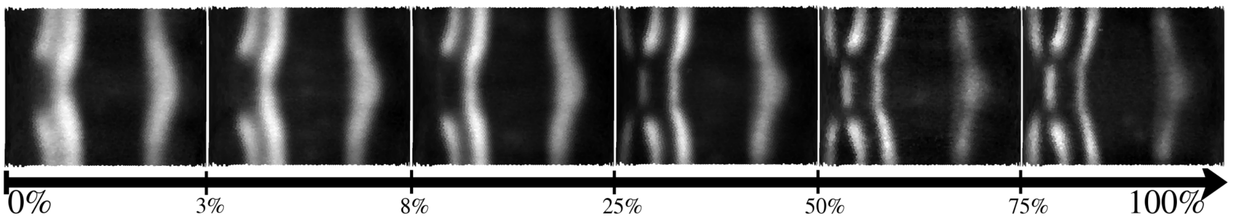


Figure 4.27: The expression pattern of *giant (gt)* shown at six different time cohorts of stage 5 of embryo development.

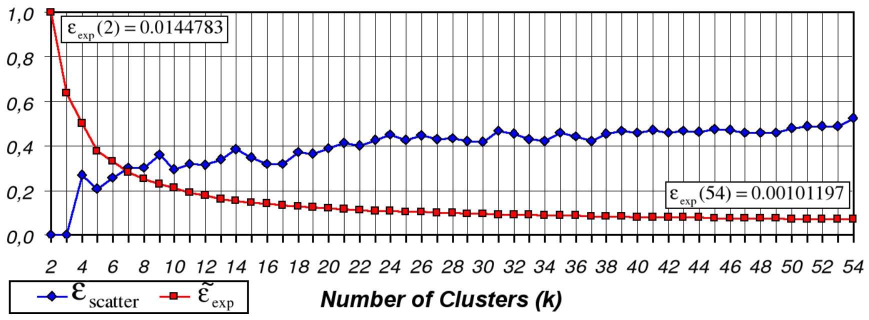


Figure 4.28: Cluster evaluation functions $\tilde{\varepsilon}_{\mathrm{exp}}$ (red) and $\varepsilon_{\mathrm{scatter}}$ (blue) for the clustering of the six time steps of *gt* ($w = 10, m = 54$). The suggested value for k is seventeen. $\varepsilon_{\mathrm{scatter}}$ further indicates that seventeen is the highest k for the particular level of detail with relatively low overall physical cluster scattering.

4.7.3 Temporal Variation Analysis

Gene expression patterns are not static but highly dynamic. Knowledge of the temporal profile of a gene expression pattern is essential for the understanding of the complex relationships between genes. Even though visual inspection of an expression pattern at different time steps provides an impression of the general temporal behavior of a gene, many important features, such as groups of cells with a similar temporal expression profile, are not easily detected and visual quantification of temporal change is not accurate. For example, the pattern of *giant (gt)* expression can be seen to change between six time cohorts within one hour, but it is not possible to rigorously describe how (see Figure 4.27). To show how PointCloudXplore (PCX) can assist in the analysis of the spatio-temporal expression pattern of genes, clustering is used to classify cells into groups of similar temporal behavior.

In Figure 4.28, the curves of the cluster evaluation functions $\varepsilon_{\mathrm{scatter}}$ and $\tilde{\varepsilon}_{\mathrm{exp}}$ are shown. The suggested number of clusters k is seventeen, which is also a local minimum of $\varepsilon_{\mathrm{scatter}}$ with $\varepsilon_{\mathrm{scatter}}(17) \approx 31.88\%$. The overall behavior of $\varepsilon_{\mathrm{scatter}}$ indicates that $k = 17$ is the largest value of k at the particular level of detail for which $\varepsilon_{\mathrm{scatter}}$ is still relatively low. A comparison

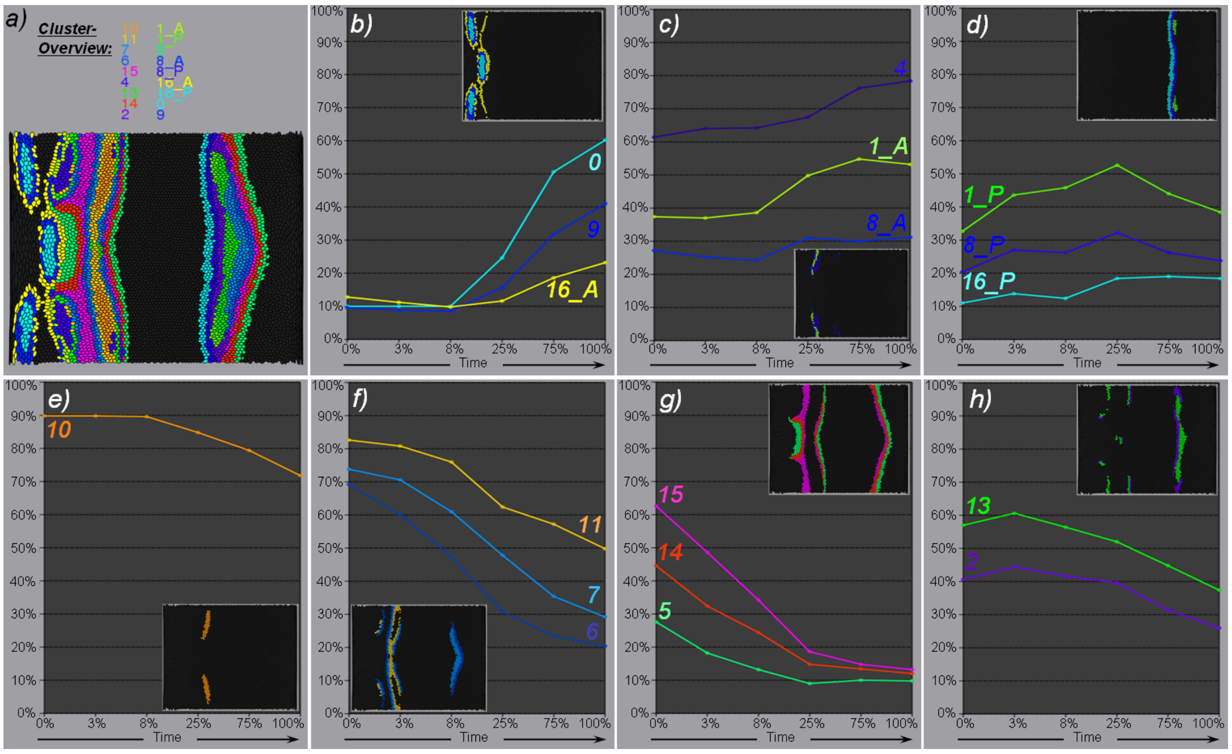


Figure 4.29: Based on the patterns of *gt* shown in Figure 4.27, cells were classified into seventeen clusters as suggested by $\varepsilon_{\text{scatter}}$ and ε_{exp}. Two clusters selected cells showing only background expression of *gt* at all time steps and are therefore not shown here. Clusters 1, 8, and 16 were each split into their two main spatial components. The remaining clusters were not split, since no significant divergence in the temporal expression profile between their main spatial components could be identified. a) An unrolled view showing all eighteen clusters of interest. b-h) The user grouped the eighteen temporal clusters into seven main groups based on their average temporal expression profiles in *gt*. The six time steps are shown on the x-axis and the expression level on the y-axis of each plot. The spatial patterns defined by the clusters are displayed in the accompanying unrolled view plots.

of $\varepsilon_{\text{scatter}}(17)$ to the next two lower local minima of $\varepsilon_{\text{scatter}}$ — with $\varepsilon_{\text{scatter}}(12) \approx 31.21\%$ and $\varepsilon_{\text{scatter}}(10) \approx 29.25\%$ — shows only a moderate increase in $\varepsilon_{\text{scatter}}$. When comparing $\varepsilon_{\text{scatter}}(17)$ to the $\varepsilon_{\text{scatter}}$ values of the next two larger local minima of $\varepsilon_{\text{scatter}}$ — with $\varepsilon_{\text{scatter}}(19) \approx 36.34\%$ and $\varepsilon_{\text{scatter}}(22) \approx 40.14\%$ — a significantly higher increase in relative physical cluster scattering is visible. This behavior can be interpreted as an indication that $k = 17$ may also provide a good compromise between a high-level and low-level description of the temporal variation of the *gt* expression pattern. A level of $k = 17$ was also confirmed to be appropriate by users of PCX.

Figure 4.29 shows as an example the result for *gt*, in which its expression patterns at six successive time cohorts were classified into seventeen clusters using k-means clustering and Euclidean distances. Two of the seventeen clusters selected cells showing only background expression at all time steps and are not shown. Each of the other fifteen clusters show distinct average expression profiles (the differently colored lines plotted in Figure 4.29), though some clusters show profiles that are closely related. In the figure, the user has grouped these clusters into seven main sub-groups based on their temporal average expression profiles, shown in panels b-h. In addition, clusters 1, 8 and 16 have each been split into two components to separate their anterior and posterior components.

Several trends can be readily seen from the different views of the analysis. The unrolled physical views show that clusters with similar average temporal expression profiles frequently, but not always, are adjacent to one another in the embryo. Expression within a set of clusters in the very anterior of the embryo increases particularly during the later time cohorts (visible, for example, in Figure 4.29b). Expression in the posterior margins of both of the major early *gt* stripes drops rapidly over the time series (Figure 4.29f-h). It is known that the location of the posterior *gt* stripe moves anteriorly during this time series [145, 99], but the data show a much more complex pattern of temporal change than has been observed previously. These results suggest that a complex combination of regulatory interactions drives these patterns.

4.7.4 Multiple Pattern Analysis

To dissect the complex regulatory interactions between genes, the expression patterns of different transcription factors that potentially act together as regulators may be used as input to cluster analysis. Cells are classified into clusters that have similar combinations of expression for the input set of regulators. Each cluster thus describes one potential sub-pattern that a regulatory network composed of these factors could give rise to. The total number of clusters then gives an approximation of the maximal complexity of the output of the network. The results of such a clustering can also be compared to the expression patterns of suspected target genes to assess possible regulatory relationships.

To provide an example of such multi-gene clustering, this section describes an examination of the relationship between the three transcriptional regulators *giant (gt)*, *hunchback (hb)*, and *Krüppel (Kr)* and the second stripe of the *eve* gene. These three factors are well-characterized regulators of this expression stripe; *hb* is an activator and *Kr* and *gt* are repressors [92]. As discussed in Section 4.7.2, the seven stripes of *eve* form characteristic clusters in gene expression space with respect to *gt*, *hb*, and *Kr* expression. By using these three factors' expression patterns as input to a clustering analysis, one can identify the potential expression pattern components that can be defined based on these regulators (see Figure 4.30). The mRNA expression values of

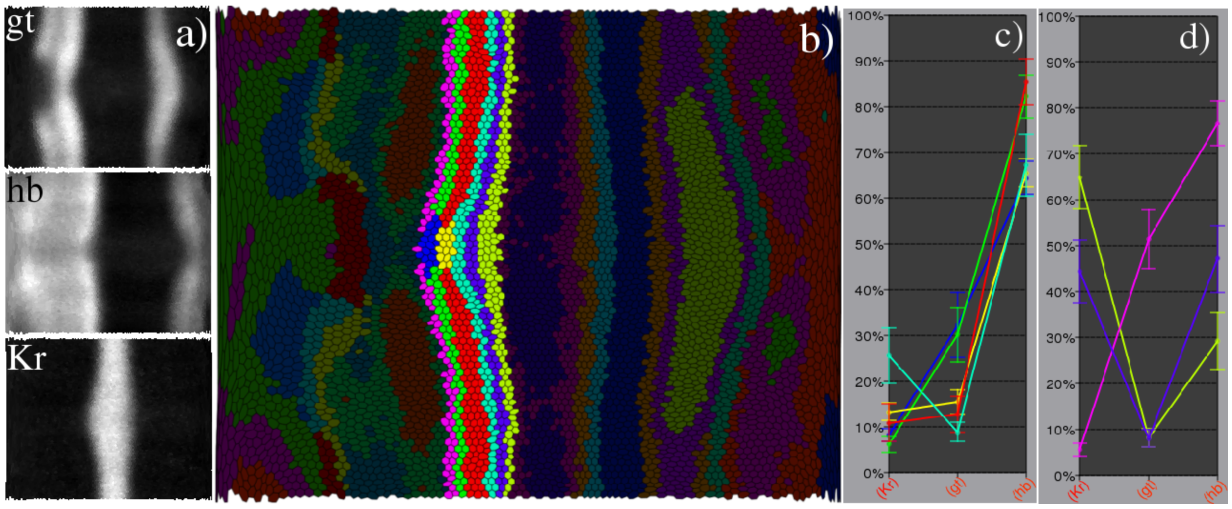


Figure 4.30: a) The transcription factors *gt*, *hb* and *Kr* at stage 5:0-3% are used as input to the clustering; their potential target is *eve* stripe 2 at stage 5:9-25% (see Section 4.7.2). b) Cells were classified into 22 clusters of which eight are of particular interest. Five clusters actually model *eve* stripe 2 and three define the inter-stripe region between stripes 1 and 2, and stripes 2 and 3. Cluster filtering was applied to three single cells only. Clusters were split in order to separate the stripe-like clusters with similar expression profiles from other spatially distant sub-clusters in the anterior and posterior region of the embryo. c) An average curve plot of the five clusters within *eve* stripe 2 showing the characteristic expression profiles of *Kr*, *gt*, and *hb*. d) Average expression curves for the three inter-stripe clusters. In both average curve plots, *Kr*, *gt*, and *hb* are shown on the *x*-axis and the level of expression along the *y*-axis.

gt, *hb*, and *Kr* from the first temporal cohort (0%-3% invagination) are here used to simulate their protein expression values at the third temporal cohort (9%-25% invagination) — the stage of the *eve* comparison target. This lag was found, on average, to be optimal for all regulators [161]. In the example, cells are classified into 22 clusters that map to locations throughout the embryo. Eight of these clusters are of interest to the control of *eve* stripe 2, five that lie within the stripe and three in the flanking inter-stripe regions. The five clusters within stripe 2 define the center, the anterior and posterior borders, as well as a ventral portion of the stripe, suggesting that these characteristic parts of stripe 2 may be different (see Figure 4.30b).

To validate the structure formed by the clusters against the target pattern, cluster colors are mapped onto an expression surface of *eve*, in which height shows the level of expression (Figure 4.31). It can be seen that the five clusters fit closely to the expression pattern of the target stripe 2.

Based on the average expression curves, the characteristic expression pattern of the potential regulators in the eight clusters that are within and flanking stripe 2 are easily visible (see Fig-

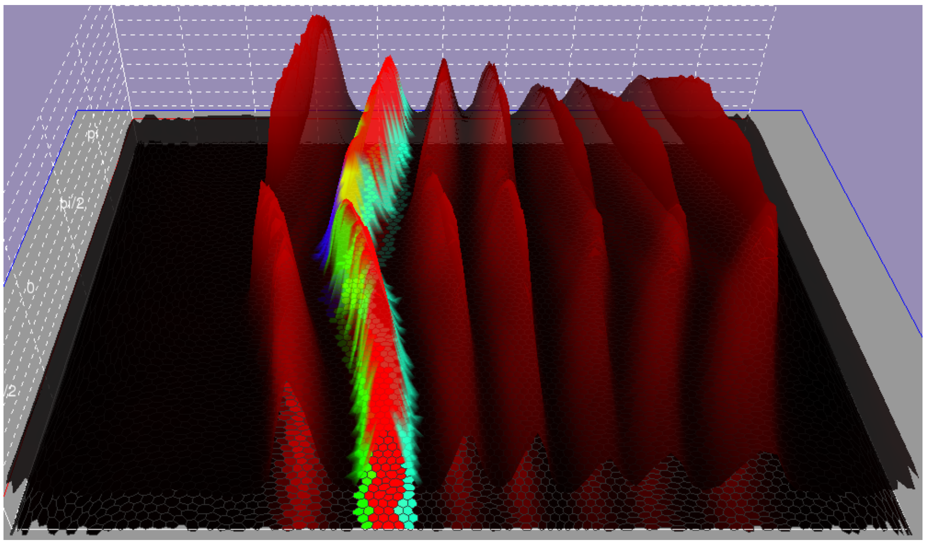


Figure 4.31: To validate the structure formed by the five clusters against the target, cluster colors are mapped onto an expression surface of *eve* where surface height shows the level of *eve* expression. The visualization shows that the clusters and the target stripe fit closely.

ure 4.30c, d). Here, *hb* is expressed at high levels in all clusters except those posterior of stripe 2, consistent with its known role as an activator of stripe 2. *Kr* is expressed at high levels only posterior of stripe 2 and *gt* is expressed at high levels only anterior of stripe 2, consistent with their known roles as repressors that define the posterior and anterior borders of stripe 2, respectively.

Interestingly, the two clusters that form short ventral patches on *eve* stripe 2 (yellow and blue) show significantly lower expression of *hb* than the two clusters that lie dorsally to them (red and green) (see Figure 4.30c, d). This correlates with a lower level of *eve* expression in this ventral margin (Figure 4.31) and suggests that this reduced expression may be the result of lower activation by *hb*. *hb* is typically thought of as regulating gene expression only along the anterior/posterior axis of the embryo. The cluster analysis suggests that it may also be able to mediate differential transcription along the dorsal-ventral axis. However, if one were to add a dorsal-ventral gene, such as *snail (sna)* (see Figure 3.3) into the analysis, it would be difficult to distinguish if the ventral gap in *eve* stripe 2 resulted from direct inhibition by *sna*, if *sna* acted via inhibiting ventral *hb* expression, or if all three expression patterns are parallel manifestations of dorsal-ventral patterning systems, each acting separately. Thus, cluster analysis can be used for identifying interesting correlations that might result from novel biological interactions or phenomena, but the analyses should be confirmed by experimental data.

This case study illustrates that clustering the expression patterns of multiple regulators can provide confirmation and additional insights into known regulatory interactions. It is likely that the extension of this strategy to less well-characterized systems will suggest potential regulatory interactions that can then be tested by other means. Furthermore, the analysis described here — based on a combination of data clustering and visualization — enables a more detailed and accurate analysis of the regulation of gene expression patterns than a manual investigation process based on the visualization alone (see Section 4.7.1).

4.7.5 Genetic Regulatory Network Modeling Using MATLAB

In Section 4.7.4 data clustering was used to identify the potential sub-patterns a regulatory network composed of a set of genes could give rise to. In such an analysis data clustering does not assume any particular genetic regulatory network model but seeks to identify all sub-patterns any such network consiting of the given input factors could give rise to. This approach allows a researcher to effectively test and define hypothesis about potential regulatory interactions. However, using this method it is not possible to investigate a specific model of a genetic regulatory network.

To overcome this limitation one can use MATLAB to compute the expression pattern a specific genetic regulatory network would define. In Sections 4.7.1 and 4.7.4 an example model for the second stripe of the expression pattern of *eve* was described with *hb* acting as an activator (+) and *Kr* and *gt* acting as repressors (-). To test this model, MATLAB is used to compute the derived expression pattern defined by the combination of these input factors.

Figure 4.32 illustrates the used genetic regulatory network model and the resulting derived expression pattern. In particular the gene-weights of the shown model are based on an empirical, user-defined model and may not be accurate. The user assigned larger weights to the suspected inhibitors of *eve* stripe 2 to enforce sharp boundaries at the anterior and posterior edges of the stripe. Furthermore, *gt* is assigned a larger weight because the anterior component of the *gt* expression pattern largely overlaps with high *hb* expression — the suspected activator of *eve* stripe 2 — whereas the expression pattern of *Kr* and *hb* do not overlap.

The resulting derived expression pattern shown in Figure 4.32 shows a distinct stripe of high derived expression at approximately the same location as the target *eve* expression pattern. Using the methods described in Section 4.7.2, the derived expression pattern is divided into its main spatial components. The clusters describing the stripe feature in the derived expression pattern are then mapped onto an expression surface of the target expression pattern (see Figure 4.33a). It can be seen that the four clusters fit closely to the expression pattern of the target stripe 2. As the cluster analysis indicated (see Section 4.7.4), the combination of *gt*, *hb*, and *Kr* may also be able to reproduce the ventral gap of lower *eve* expression. The tested model shows a similar

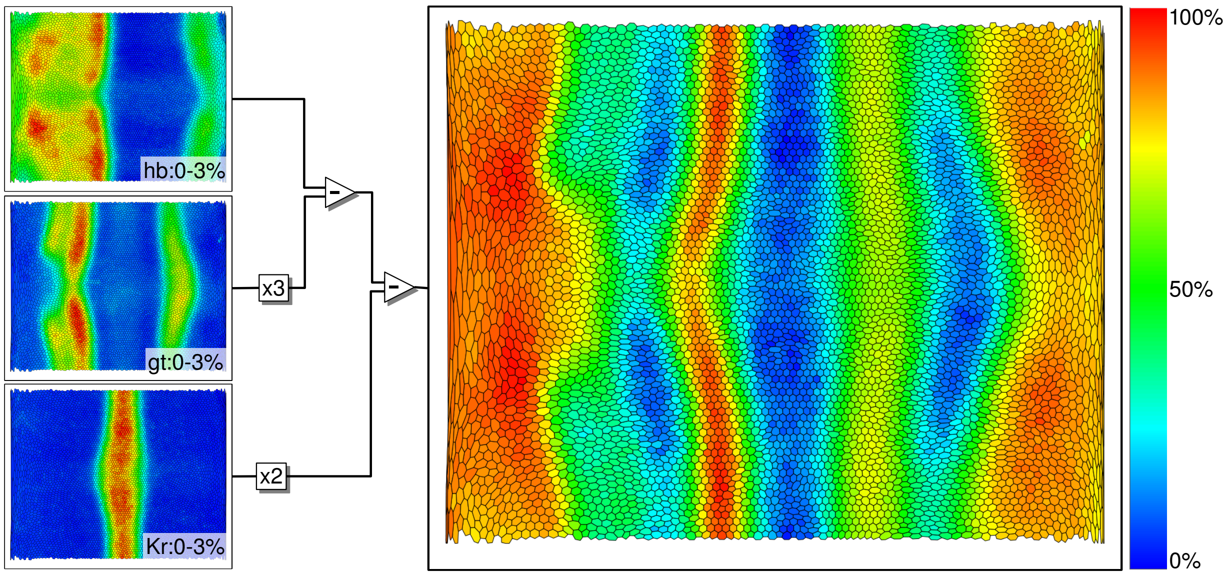


Figure 4.32: Empirical model for the regulation of *eve* stripe 2. The expression patterns of *gt*, *hb* and *Kr* at stage 5:0-3% (left) are combined defining a new derived expression pattern (right). At approximately the same location as *eve* stripe 2 a distinct stripe of high derived expression appears.

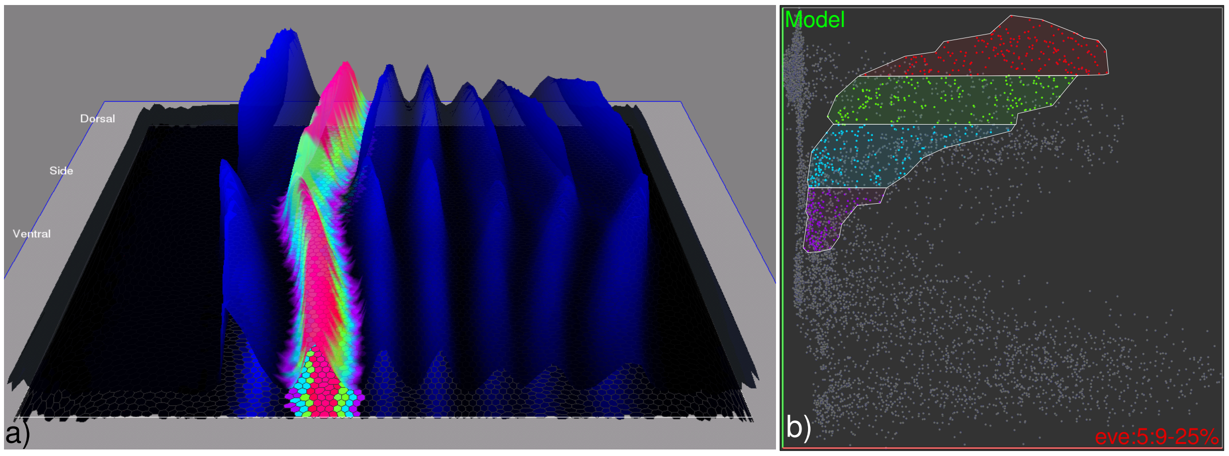


Figure 4.33: To validate the empirical model shown in Figure 4.32, the stripe-like feature in the modeled expression pattern is extracted using data clustering and cluster splitting. Four clusters are created, each representing a different expression level in the modeled pattern within the according region (lilac/blue=medium , green/red=high). a) To validate the structure formed by the five clusters agains the target, cluster colors are mapped onto an expression surface of *eve* at stage 5:9-25%. The visualization shows that the clusters and the target stripe fit closely. b) Scatter plot of *eve* and the model ($m = hb - 3 * gt - 2 * Kr$). The clusters define a characteristic shape in the scatter plot indicating a possible correlation between the model and the target. The outline of the clusters were added by hand to highlight their shape.

behavior with high derived expression (red) appearing only dorsally of a patch of lower derived expression (green) on the ventral surface of the embryo.

Figure 4.33b shows a scatter plot of *eve* and the modeled derived expression pattern illustrating the correlation between the model and the target pattern. The clusters defining the stripe in the derived expression pattern form a characteristic shape in the scatter plot indicating that the model and the target may be correlated.

4.7.6 Temporal Variation Analysis and Cluster Validation Using MATLAB

Quantification of the temporal change of an expression pattern based on visual inspection is inaccurate. Furthermore, subtle temporal changes are not easily detected via visual comparison of plots from different time steps. To aid in this process MATLAB can be used to compute differential expression patterns describing the change in expression between two time steps. Inspection of differential patterns allows visual detection of even subtle changes in expression.

Figure 4.34 shows an example analysis of the temporal variation of the gene *giant (gt)* based on differential expression patterns. By comparing the *gt* expression pattern at six time cohorts within one hour the pattern can be seen to change (see Figure 4.34 top). Subtle changes between consecutive time steps are however not easily identified. Inspection of the differential expression patterns of *gt* computed using MATLAB and visualized in PointCloudXplore (PCX) enables a much more detailed analysis of the temporal variation of the *gt* expression pattern (see Figure 4.34 bottom). For example, while the pattern of *gt* at stage $5:[0-3\%]$ and $5:[4-8\%]$ ap-

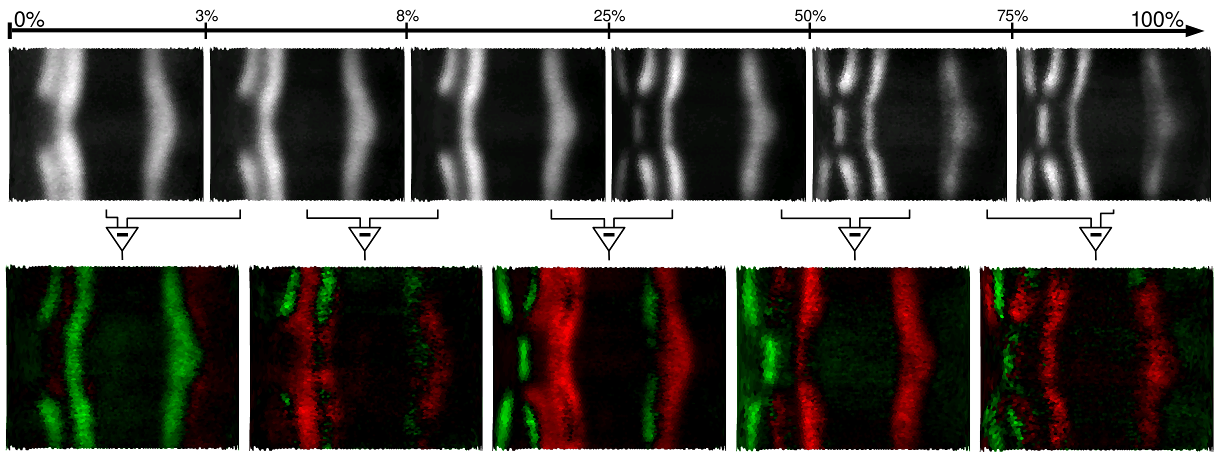


Figure 4.34: Differential change of the expression of *gt* over time. Top: Expression pattern of *gt* at six time steps during stage 5 of embryo development. Bottom: Differential change between the consecutive time steps. Color indicates positive (green) and negative (red) change in expression.

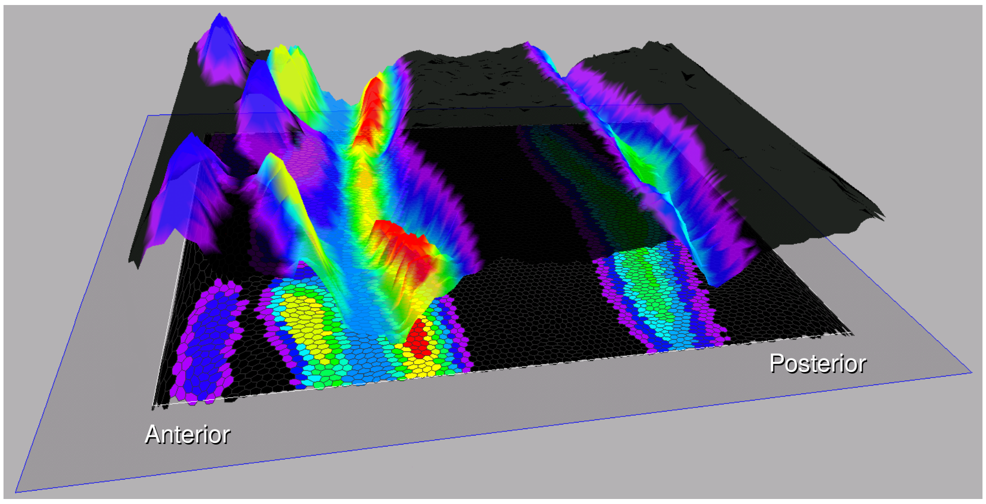


Figure 4.35: Expression surface of the differential expression pattern $gt_{diff} = gt_{[5:75-100\%]} - gt_{[5:0-3\%]}$. colored according to 17 clusters describing the temporal variation of the *gt* expression pattern (see also Section 4.7.3). Two clusters that did not show any significant expression of *gt* at all times are shown in black. The clusters identify the main temporal changes of the pattern indicating that the clustering result is valid.

pear to be similar the differential pattern reveals significant changes in expression in the anterior as well as posterior expression domain of *gt*.

In Section 4.7.3 clustering was used to characterize the temporal variation of the the gene *gt*. A valid temporal clustering should capture the temporal changes of the input pattern well. Figure 4.35 shows the results of a temporal clustering of the gene *gt* mapped onto an expression surface describing the difference in *gt* expression between the first and last time step. The clusters coincide with the main characteristic features (hills/valleys) of the differential expression pattern of *gt* indicating that the clustering result is valid.

Besides for validation of temporal clusterings, differential patterns can also be used in the clustering itself. A temporal clustering based on the measured patterns of a gene allows the identification of cells that show similar expression of a gene over time. By clustering cells based on differential patterns instead, groups of cells that undergo similar changes in expression over time can be identified. Using both, differential and original expression patterns in the same clustering promises to improve the quality of a temporal clustering experiment.

Chapter 5

Visualization and Analysis of Laser Wakefield Particle Acceleration Data

Analysis and knowledge discovery from large, complex, multi-variate laser wakefield particle accelerator (LWFA) simulation data is a challenging task. One main feature researchers are interested in are beams of high-energy particles formed during the coarse of LWFA simulations. To enable efficient and accurate analysis of these particle beams, dedicated mechanisms for selection and detection of particle beams are needed. Furthermore, to support analysis of selected particle bunches of interest effective visualization methods are required. To address these challenges, this work presents a novel framework for knowledge discovery from LWFA simulations based on state-of-the-art data management, high-performance visual data exploration, and automatic data analysis.

To enable visual analysis of particle beams in extremely large and complex scientific data this work describes the integration of the advanced visualization system VisIt and state-of-the-art data management using FastBit. This approach, while applied here to accelerator science, is generally applicable to a broad set of science applications, and is implemented in a production-quality visual data analysis infrastructure.

To enable more efficient and accurate analysis of particle beams and to support analysis of a larger number of datasets, this work describes two novel methods for automatic beam analysis. The automatic beam detection algorithm is based on a combination of bunch lifetime estimation and fuzzy clustering and is aimed at identification of the highest-energy particle bunch in a simulation. The automatic beam path analysis on the other hand has the goal to enable efficient temporal classification of multiple particle bunches based on the complete temporal history of the particles that form them. Automatic analysis of particle beams based on temporal particle paths enables accurate classification of particle beams and supports analysis of the temporal beam evolution.

As high-level novel scientific contributions of this work this chapter discusses:

(i) A novel framework for high-performance visual data exploration of extremely large data

(ii) Automatic beam detection; a novel approach for automatic detection of the highest-energy particle bunch based on bunch lifetime analysis and fuzzy clustering

(iii) Automatic beam path analysis; a novel approach for automatic detection and analysis of multiple particle beams based on the complete temporal path of the particles that form them

(iv) Integration of the automatic analysis with visualization to enable efficient analysis of even extremely large 3D particle datasets

(v) The use of the proposed framework to address relevant questions, such as, (a) detection and definition of particle beams, (b) analysis of the formation and acceleration of particle beams, (c) investigation of the quality of particle beams, and (d) comparison of the behavior and quality of multiple particle beams.

5.1 Problem

Laser wakefield accelerators (LWFAs) promise to be a new compact source of high-energy particles and radiation, with wide applications from medicine to physics. Development and optimization of LWFAs is challenging task. Laboratory experiments are expensive and require careful planning. Furthermore, it is in practice not possible to record the complete evolution of LWFA experiments but only to measure their outcome, such as, the produced radiation or particle beam. Physicists, therefore, model the acceleration process computationally via particle-in-cell (PIC) simulations (see Section 3.2) to gain insight into the complex physical processes in LWFAs.

LWFA simulations model the effects of the interaction between a laser pulse and a hydrogen plasma. Similar to the wake of a boat, the radiation pressure of the laser pulse displaces the electrons in the plasma. Together with the space-charge restoring force of the ions, this displacement drives a wave (wake) in the plasma. If the wake has a high enough amplitude then electrons can become trapped and accelerated by the plasma wave, and eventually decelerate again as they outrun the wake.

Analysis of LWFA simulations is a challenging task; with the produced datasets being: (i) extremely large, (ii) of varying spatial and temporal resolution, (iii) heterogeneous, and (iv) high-dimensional. Due to the large amount of memory required to save all the information of a single timestep, only a small subset of all simulated timesteps is actually saved to file. In contrast to the actual simulation, the temporal resolution of the stored data may, therefore, not be sufficient to resolve wave frequencies. Even when only a small number of timesteps is saved (≈ 30), the produced data may still be extremely large. In the case of 3D simulations the total size of a single simulation dataset is usually in the order of several terabytes (TBs). The amount of data created by a single simulation run depends on various parameters, e.g., (i) the spatial resolution of the simulation (i.e., number of particles per cell and resolution of the grid used in the simulation), (ii) the temporal resolution of the data (i.e., how many timesteps are written to file), and (iii) size of the simulation window (i.e., the area around the laser pulse captured by the simulation). All these are user-defined parameters; with increasing performance of supercomputers and decreasing cost for storing simulation results, the size of datasets is expected to increase even further in future. Furthermore, LWFA simulation data is high-dimensional and heterogeneous with, e.g., electric and magnetic field data being defined on a grid and particles being scattered data.

Methods developed for the analysis of LWFA simulation data must therefore be efficient and deal robustly with data of varying temporal and spatial resolution. The analysis should furthermore be flexible and allow for incorporation of user knowledge. In the context of high-end numerical simulations it is crucial that the visual analysis does not become the bottleneck for simulation output, hence, the visualization should require a minimum of user intervention [132]. With researchers producing increasingly large collections of simulation data the need arises for automating the most time-consuming steps of the data analysis process.

One main feature physicists are interest in are beams of highly accelerated particles formed during the coarse of a simulation. In order to gain a deeper understanding of the complex acceleration process, an effective analysis needs to address complex questions such as: (i) which particles become accelerated; (ii) how are particles accelerated, and (iii) how was the beam of highly accelerated particles formed and how did it evolve [106]. Furthermore, the analysis needs to support: (iv) investigation of the quality of particle beams, and (v) comparison of the behavior and quality of multiple particle beams. To effectively address these question the analysis must support accurate and efficient selection of particle beams and fast tracing of selected features (i.e., groups of particles) through time.

The remaining parts of this chapter are structured as follows. Section 5.2 first describes the analysis pipeline currently employed by accelerator scientists at the LOASIS [111] program at LBNL. The proposed novel framework for knowledge discovery from LWFA simulations is introduced in Section 5.3. Afterwards, the different parts of the proposed analysis framework are described in detail in Sections 5.4 to 5.7. This chapter concludes with a presentation of results and applications of the proposed analysis framework (see Section 5.8).

5.2 State of the Art: Analysis of Laser Wakefield Particle Acceleration Data

Before describing the proposed framework for knowledge discovery from LWFA simulations (see Section 5.3 to 5.7), this section introduces the data generation and analysis pipeline currently employed by scientists at the Laser Optics and Acceleration System Integrated Studies (LOASIS) program headed by Dr. Wim Leemans. LOASIS is a core program within the Accelerator and Fusion Research Division (AFRD) at the Lawrence Berkeley National Laboratory (LBNL). The work presented in this chapter was conducted in collaboration with researchers from the LOASIS program.

Figure 5.1 provides an overview of the data generation and analysis pipeline employed by physicist at the LOASIS program. Simulation, theoretical study of the fundamental physics, and laboratory experiments are used together to enable detailed study and development of LWFAs. Theoretical study is essential to enable computational modeling of experiments and provides understanding of the fundamental physics of plasma-based acceleration. Simulations allow the study of phenomena that cannot be recorded in real-world experiments or accessed by theory. Laboratory experiments on the other hand are important to validate and verify theoretical and computational models, identify problems, and proof that a particular LWFA design is functional and practical.

In the context of todays LWFA experiments, simulations are essential for the understanding of the fundamental physics of plasma-based acceleration, understanding of the processes and results observed in experiments, as well as for improvement of experiments. With increasing accuracy and reliability of computational models and decreasing simulation costs, simulation also becomes increasingly important for planning of experiments and testing of new accelerator models. Simulation is, therefore, becoming an increasingly important tool to identify good physical parameters for experiments as well as to build new hypothesis to be verified via experimentation. Simulation also promises to be an invaluable tool in the design process of new accelerators allowing researchers to test and validate different designs, hence, reducing risk (cost and safety) as well as enabling improvement of the accelerator before construction. In laboratory experiments it

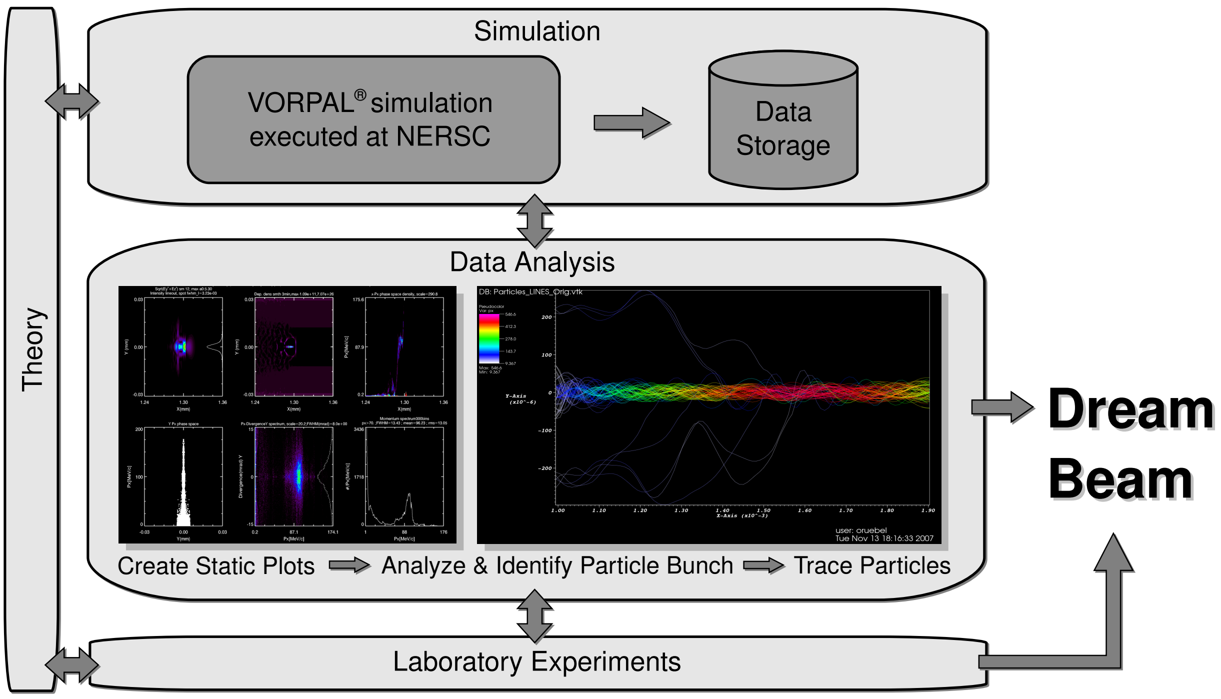


Figure 5.1: Laser wakefield particle acceleration data generation and analysis pipeline. Simulation, theoretical study, and laboratory experiments are used together to enable detailed study and development of LWFAs. Both, simulation and experiments, are linked via data analysis and theory enabling improvement of experiments based on simulation results and vice versa. The example data analysis pipeline shown here refers to the analysis of the simulation data.

is usually not possible to capture the whole evolution of an experiment but only its outcome, e.g., via measurements of the produced radiation or particle beam. Furthermore, complex processes, such as, (a) nonlinear plasma response, (b) beam trapping, (c) self-consistent laser propagation, and (d) beam acceleration, are not readily accessible to analytic theory. Simulation enables *in silico* study of these complex physical phenomena greatly improving the understanding of the fundamental physics of laser wakefield acceleration.

Laboratory testing and experimentation is the main tool for verification and validation of simulation results. Numerical simulations of LWFAs can only approximate the underlying physical phenomena relevant for particle acceleration and therefore require verification and validation. Based on the findings in laboratory experimentation, hypotheses derived from simulation are verified, simulation algorithms are validated and improved, as well as potential errors and problems are identified. Furthermore, not all physical phenomena relevant for the study of LWFAs are accessible to simulation today and, hence, need to be studied through experimentation (see Section 3.2). Laboratory experiments are also essential to investigate the effectiveness of a particular

LWFA design for use in real-world applications, e.g., to build a particle collider.

The ultimate goal of LWFA research is to develop an LWFA that produces a “Dream Beam,” i.e., a high-energy particle beam with very low energy spread (see Figure 5.1). Besides just proof of concept, the accelerator should also be reliable, i.e., predictably produce high-quality beams. To achieve this goal, many iterations through simulation, theoretical study, and laboratory experimentation are necessary to optimize physical parameters and the accelerator design, to identify problems, and gain insight into the complex acceleration process. Similar to laboratory experiments, the *in silico* analysis based on simulation is itself an iterative process in which simulation parameters are iteratively refined and new simulations executed.

Data analysis is essential for the understanding of LWFA simulation and experimental data. Theoretical study and data analyses also define the main links between the simulation and laboratory experiments enabling improvement of lab experiments based on simulation and vice versa. In particular in the context of high-end numerical simulations, it is crucial that the visual analysis does not become the bottleneck for simulation output and knowledge discovery.

Analysis of large, complex LWFA simulation dataset is a challenging task. Researchers at the LOASIS program currently use IDL-based [129] [1] visualization and analysis scripts for data analysis. After completion of a simulation, researchers execute a set of these scripts to create different visualizations (usually six plots per image) for each timestep of the simulation. These images are then combined to make animations of the whole time series. Figure 5.2 shows an example image of a single timestep of such an animation consisting of six different plots. Using Perl scripts, scientists typically create several different sets of animations for each simulation, each showing a different set of plots of the data. The data is commonly displayed using various types of two-dimensional plots. The most common types of plots used for data display are 1D or 2D density plots (histograms), iso-contour plots, and pseudocolor plots. In particular for investigation of 3D simulations scientist are now also starting to VisIt. Another main analysis tool, besides visual investigation of the data, is statistical analysis. Using dedicated scripts, researchers derive a series of statistical measurements from the simulation data such as average, root-mean-square, and standard deviation values of different variables, measurements of the laser energy and many more. The combination of visual inspection of the data and statistical analysis allows researchers to effectively study the general evolution of simulations.

In order to analyze particle beams and their temporal history, a scientist investigates the animations and static views of a simulation to: (i) identify an appropriate timestep at which a beam of interest exists, and (ii) identify thresholds to be able to extract the relevant particles from the

[1] *IDL = Interactive Data Language.* IDL is an array-oriented programming language with various features for mathematical analysis and graphical display and is commonly used for data analysis.

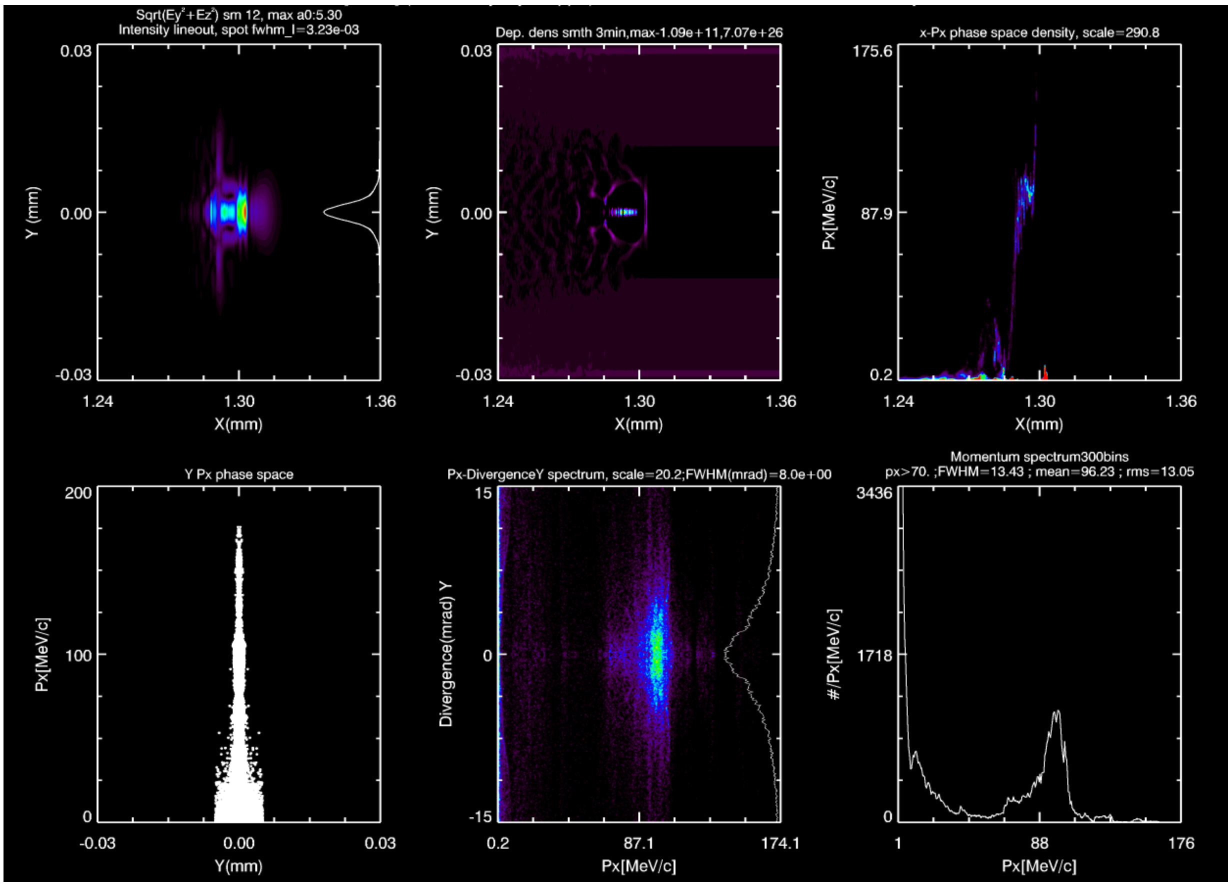


Figure 5.2: Example showing a single frame of a movie consisting of a set of standard plots used by laser wakefield accelerator scientists for data understanding and analysis. Top from left to right: (a) Plot of the laser envelope, (b) Plot of the plasma density in physical space (x,y), and (c) Plot of beam phase space (x,px). Bottom from left to right: (d) Beam scatter plot of (y,px) space, (e) Plot of the beam momentum-divergence spectrum, and (f) Plot of the beam spectrum. *(Image courtesy of Cameron G. R. Geddes)*

data. Commonly, a single threshold in px is used to define the beam. The according parameters — timestep and threshold(s) — serve as input to yet another script for particle tracing. The tracing script, implemented in IDL, first identifies the IDs of the particles of interest. Afterwards, the script performs a sequential scan through the data to extract the relevant particles from each timestep. Particle tracing is a time-consuming process and, hence, impractical as an every-day analysis tool. Using the described scripts it takes, e.g., ≈2.5 hours to track 250 particles for a small 5GB dataset. Furthermore, owing to IDL limitations, the scripts cannot load all necessary data for extremely large 3D datasets but require incremental data access, which in turn leads to a further decrease in performance.

Having introduced the problem and general analysis process, the next section now provides an overview of the proposed novel framework for knowledge discovery from LWFA simulation data. Afterwards the different parts of the framework are described in detail in the then following sections.

5.3 Overview

While statistical analysis and visual inspection provide effective means for understanding the general characteristics of a simulation, the approach is limited in particular with respect to the analysis of specific features of the data. As described earlier, in the context of laser wakefield acceleration, scientist are, e.g., interested in particle beams — i.e., compact bunches of highly accelerated particles — formed during the coarse of a simulation. Such particle beams constitute small features of the data. Definition and analysis of particle beams based on static plots or animations is time-consuming, non-interactive, and error-prone. To overcome these limitations this work describes a novel approach towards visual exploration of extremely large scientific data. Using this new approach a researcher can interactively specify and trace particles over time leading to a more responsive and accurate data analysis process. Using the proposed system a user receives immediate feedback about important characteristics of a selection enabling fast validation and hence more accurate selection of data features. Also, the proposed approach allows a user to more easily define selections based on multiple variables as well as information from different timesteps enabling not only more accurate selection but also selection of more complex features than possible when using only a single threshold in *px*. Using efficient data management, fast selection as well as fast tracing of particles over time becomes possible. With the increased performance, particle tracing becomes practical as an every-day analysis tool. Scientists can, hence, more easily trace multiple features over time, refine features based on information from the tracing, and repeat the tracing several times. While manual feature selection and detection allows for detailed analysis of particle beams it may be impractical for the analysis of a large collection of datasets due to the substantial manual input that is required. To assist researchers in the analysis of an increasing number of datasets this work also introduces several methods for automatic detection and analysis of particles beams.

Figure 5.3 provides an overview of the proposed framework for knowledge discovery from laser wakefield accelerator (LWFA) simulation data. LWFA simulation datasets are extremely large — in the order of several terabyte (TB) — and are expected to become even larger in future. As mentioned earlier, in practice a researcher is often only interested in a small subset of

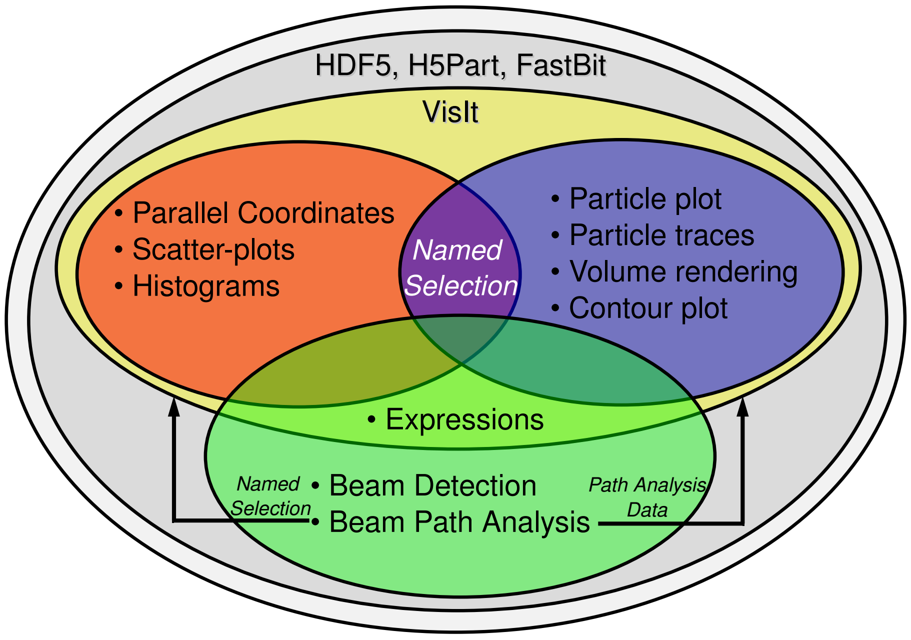


Figure 5.3: Analysis framework for knowledge discovery from laser wakefield particle accelerator simulation data.

the data, e.g., in this case beams of highly accelerated particles. Using efficient data management based on HDF5[2], H5Part[3] and FastBit[4] efficient analysis and access of selected data portions of interest becomes possible.

The state-of-the-art, parallel visualization and graphical analysis system VisIt[5] is used as main tool for visual exploration of the data. VisIt contains a rich set of visualizations ranging from information visualization views, such as scatter-plots and parallel coordinates, to advanced scientific visualizations, such as particle, volume, or contour plots. VisIt also integrates several data analysis tools and supports, e.g., also data analysis based on derived quantities via the concept of so called expressions. Within VisIt different visualizations and views can be linked in various ways, e.g., by synchronizing data operators, or locking time and camera view.

By integrating FastBit and VisIt, this work enables interactive analysis of extremely large particle datasets. Using FastBit, VisIt can efficiently evaluate data queries and extract data por-

[2]HDF5 is developed by the HDF5Group and is available online at: `http://www.hdfgroup.org/HDF5/`. HDF5 is also available directly as part of the VisIt installation.

[3]H5Part is developed by the Visualization Group at LBNL and is available online at:`http://www-vis.lbl.gov/Research/AcceleratorSAPP/` H5Part is now also available directly as part of the VisIt installation.

[4]FastBit is available online at: `http://sdm.lbl.gov/fastbit/`. FastBit is now also directly available as part of the VisIt installation.

[5]VisIt is developed by the VisIt development team and is available online at: `http://www.llnl.gov/visit/`

tions of interest from the raw data based on multi-dimensional threshold and/or ID-based queries (see Section 5.4.3). To provide the user with an effective interface for performing these data selection operations this work describes several new extension to histogram-based parallel coordinates (see Section 5.4.2). Using the concept of ID-based queries, efficient tracing of particle subsets over time becomes possible(see Section 5.4.3.2).

While interactive selection of particle beams is effective it requires substantial manual input from the user and may, hence, be time-consuming. Automating the most time-consuming parts of the data analysis process promises to support a more focused and efficient analysis process enabling the analysis of large collections of LWFA simulations. To assist researchers in the analysis of a large number of datasets this work presents, (i) *automatic beam detection*, a method aimed at identifying the highest energy particle bunch in a LWFA simulation, and, (ii) *automatic beam path analysis*, a method aimed at identification and comparison of multiple particle bunches.

Automatic beam detection, uses a combination of a bunch lifetime analysis based on a particle density estimation and fuzzy clustering to estimate spatially confined particle beams. Automating the detection of the highest energy beam promises a more focused and effective analysis process by allowing a user to concentrate on the main outcome of a simulation without having to manually specify the main feature of interest. The described method is implemented as a stand-alone application in *R* [6].

Automatic beam path analysis, on the other hand, is aimed at detection of possibly many particle bunches formed during the coarse of a simulation and classification of their temporal behavior. The goal is to enable a user to: (i) more quickly and accurately define particle bunches, (ii) more effectively analyze the temporal evolution of particle bunches, and to (iii) enable comparison of different bunches. The proposed analysis pipeline consists of several different steps. First, each timestep is analyzed independently to detect the different particle bunches. Afterwards the method derives for each main particle bunch a reference path describing its temporal evolution. Based on the distance between a particle's path and the reference path the user can then accurately and efficiently identify the particles that form a bunch. The beam path analysis is implemented as a stand-alone application in C++ using FastBit to accelerate data access.

Integration of the analysis with the visualization is done in multiple ways. The automatic beam detection directly employs the visualization capabilities of R for investigation of analysis results. The beam path analysis then creates a set of particle path files describing the classification and temporal behavior of particle bunches. The user can access and visualize these datafiles directly from within VisIt and investigate analysis results in the context of the raw data. Based

[6] R is available online at: `http://www.r-project.org/`

on the analysis results the algorithm (or the user) may then define so called named selections describing which particles form a bunch. *Name selections* are a new concept in VisIt that allows for persistent selection of particles based on their identifier (ID). In VisIt a user can apply named selections defined based on the automatic analysis to the original data for further investigation.

The main, high-level contributions of the work discussed in this chapter are:

(i) Section 5.4 describes a novel approach for visual exploration of extremely large data sets based on the integration of FastBit, VisIt, and histogram-based parallel coordinates.

(ii) A novel algorithm for automatic detection of the highest-energy particle bunch based on bunch lifetime estimation and fuzzy clustering is presented in Section 5.5.

(iii) Section 5.6 then describes automatic beam path analysis; a novel approach towards detection and analysis of multiple particle bunches in laser wakefield accelerator simulations based on the analysis of particle paths.

(iv) The integration of the analysis with the visualization is discussed in Section 5.7.

(v) This chapter concludes with a presentation of results in which the proposed framework is used to analyze particle beams in laser wakefield accelerator simulation data (see Section 5.8). The presented analysis illustrates how a user can: (a) define particle bunches of interest, (b) analyze the formation and temporal evolution of a particle bunch, and (c) investigate and compare the quality of multiple particle bunches.

In the context of the work on visual exploration of extremely large particle data the specific new contributions of this work are:

(i) A novel approach for fast generation of histogram-based parallel coordinates is presented in Section 5.4.2.

(ii) To enable fast selection and tracing of particles over time this work discusses the integration of the visualization system VisIt and the index/query system FastBit. (see Section 5.4.1 and 5.4.3).

(iii) The performance of the different parts of the system (i.e., histogram computation, data selection, and particle tracing) is examined showing fast response times and scalability of the proposed approach.

In the context of the method for automatic beam detection (see Section 5.5), the specific new contributions of this work are:

(i) A new method for identification of features of high particle density indicating the location of potential particle beams in space as well as their lifetime is described (see Section 5.5.2 and 5.5.3).

(ii) The application of fuzzy clustering for detection of potential particle beams at single timesteps is introduced (see Section 5.5.4).

(iii) By combining the results from the bunch lifetime analysis and fuzzy clustering this work enables the estimation of the highest-energy beam formed during the course of a LWFA simulation (see Section 5.5.5).

(iv) Finally, the proposed algorithm is validated using four different simulation datasets illustrating the effectiveness of the proposed approach (see Section 5.5.6).

The specific new contributions of the work about automatic beam path analysis of laser wakefield acceleration data (see Section 5.6) are:

(i) A novel pipeline for automatic detection and analysis of particle bunches in laser wakefield particle accelerator simulations is described. As part of the pipeline the following new methods are introduced:

   a) An efficient method for detection of particle bunches at single timesteps of a simulation (see Section 5.6.4).

   b) A method for combining particle bunch classifications from different timesteps to define a single consolidated description of multiple particle bunches detected in a single simulation (see Section 5.6.5).

   c) A novel method for classification of particle bunches based on the temporal paths of the particles that form them. In this process the analysis also derives additional information about the different temporal phases of particle bunches, e.g., when was a bunch formed or accelerated (see Section 5.6.6).

(ii) To achieve good performance the algorithm makes use of FastBit for efficient data access and particle tracing. In this context, this thesis introduces new functions for computation of three-dimensional (3D) conditional histograms that were specifically developed for this work and integrated into FastBit.

(iii) The proposed method is applied to a variety of 2D as well as 3D particle data sets to demonstrate its validity and effectiveness.

(iv) Section 5.6.8 examines the performance of the proposed analysis pipeline and its different components.

5.4 High-performance Multivariate Visual Data Exploration

The work described in this section focuses on combining and extending two different but complementary technologies aimed at enabling rapid, interactive visual data exploration and analysis of laser wakefield particle acceleration data. While applied here to accelerator science, the described approach is generally applicable to a broad set of science applications and is implemented in a production-quality visual data analysis infrastructure.

To support highly effective visual data exploration, knowledge discovery and hypothesis testing, this work adapts and extends the concept of parallel coordinates, in particular binned or histogram-based parallel coordinates, for use with high-performance query-driven visualization of very large data. In the context of visual data exploration and hypothesis testing, the parallel coordinates display and interaction mechanism serves multiple purposes. First, it acts as a vehicle for visual information display. Second, it serves as the basis for the interactive construction of compound Boolean data range queries. These queries form the basis for subsequent *drill down* or data mining operations.

To accelerate data mining, this work leverage state-of-the-art index/query technology using FastBit to quickly mine for data of interest as well as to quickly generate multi-dimensional histograms used as the basis for rendering of the aforementioned parallel coordinates. This combination provides the ability for rapid, multi-dimensional visual data exploration. All these methods have been integrated into the advanced visualization system VisIt.

One main scientific impact of this work is that it vastly reduces the duty cycle in visual data exploration and mining. In the past, accelerator scientists performed the tracing of particles over time using scripts that performed a search at each timestep for a set of particles. The runtime for this operation was on the order of ours. Using the proposed implementation, the runtime of the same operation is reduced from hours to seconds.

The specific new contributions of this work are:

(i) A novel approach for fast generation of histogram-based parallel coordinates is presented (see Section 5.4.2).

(ii) Via integration of the visualization system VisIt and the index/query system FastBit, this work enables fast selection and tracing of particle bunches over time (see Section 5.4.1 and 5.4.3).

(iii) The typical workflow for visual exploration of particle accelerator data based on the proposed system is described (see Section 5.4.4). A detailed use case is then presented later, in which the proposed system is used to solve a challenging scientific data understanding problem in accelerator science (see Section 5.8.1).

(iv) The performance of the proposed approach is examined using a modern HPC[7]. platform and a very large (≈1TB) complex laser wakefield particle accelerator simulation dataset

(v) With respect to VisIt this work describes in addition the following new extensions:

a) *Named selections*; a novel approach towards ID based selection in VisIt (see Section 5.4.3.1).

b) *Persistent Particles*; a new operator for efficient particle tracing in VisIt (see Section 5.4.3.2).

5.4.1 System Design

Figure 5.4 shows a high-level view of the different components and data flow of the proposed system for high-performance visual exploration of laser wakefield simulations based on VisIt and FastBit. Raw scientific data, which is produced by simulation or experiment, is augmented by the computation of indexing data. Here, this step is performed outside the visual data analysis application VisIt as a one-time preprocessing. The here described implementation uses FastBit [51] for creating index structures. The data sizes are described in more detail later in Section 5.8.1.

After the one-time preprocessing step, VisIt uses FastBit at the data-loading portion of the pipeline to quickly compute histograms and to perform high-performance data subsetting/selection based upon multi-variate thresholds and/or particle identifiers (IDs) (Section 5.4.3). The histograms serve as the basis for visual presentation of full-resolution and subset views of data vis-a-vis parallel coordinates (Section 5.4.2). In the described implementation, the computational complexity of rendering parallel coordinates plots – both context and focus views – is a function of histogram resolution, not the size of the underlying data. The approach is, therefore, particularly well suited for application to extremely large data such as laser wakefield particle accelerator simulations (see Section 5.4.2).

In the context of this work, parallel coordinates serve as the main vehicle for steering data mining operations (see Section 5.4.2). Using parallel coordinates a user can define multi-dimensional range queries while receiving immediate feedback about characteristic properties of the selection.

Once a user has identified a subset of particles of interest VisIt can store the selection in the form of a *Named Selection*. Named selections define yet another type of query of the form

[7] *HPC = High-performance computing*; The term HPC refers to the use of supercomputers and computer clusters to solve advanced computation problems. In the context of this work the massively parallel processing (MPP) system Franklin at NERSC is used as reference HPC platform

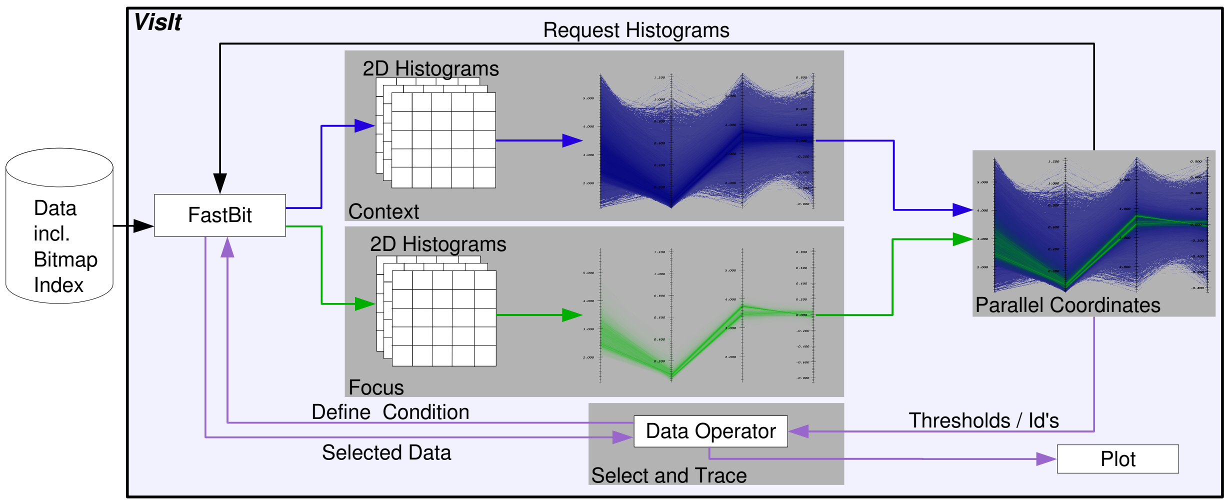


Figure 5.4: Overview of the major components and data flow paths in the proposed system for high-performance visual data exploration of extremely large data. Large-scale scientific data and indexing metadata are input via a parallel I/O layer, which allows the system to achieve high levels of performance and parallel efficiency. VisIt uses the FastBit API at the I/O layer to perform parallel computation of multidimensional histograms as well as data subsetting. Results of those queries are then used downstream in the visualization application for presenting information to the user and in support of interactive data mining actions.

$ID \in (id1, id2, ..idn)$ (a so called ID-based or *Equality*-query) (see Section 5.4.2.4). Like range queries, FastBit is also used to accelerate ID-based queries. ID-based queries allow the same particle subset to be extracted from different timesteps. By issuing a series of ID queries over the complete time series, efficient particle tracing — i.e., computation of the temporal paths of particles — becomes possible (see Section 5.4.3.2).

5.4.2 Histogram-based Parallel Coordinates

Parallel coordinates provide a very effective interface for defining multi-dimensional queries based on thresholding. Using sliders attached to each axis of the parallel coordinates plot, a user defines range thresholds in each displayed dimension. By rendering the user-selected data subset (the focus view) in front of a parallel coordinates plot created from the entire data set (the context view), the user receives immediate feedback about general properties of the selection. Data outliers stand out visually as single or small groups of lines diverging from the main data trends. Data trends appear as dense groups of lines (here indicated by bright colored bins). A quick visual comparison of the focus and context views helps to convey understanding about similarities and differences between the two (see, e.g., Figure 5.8).

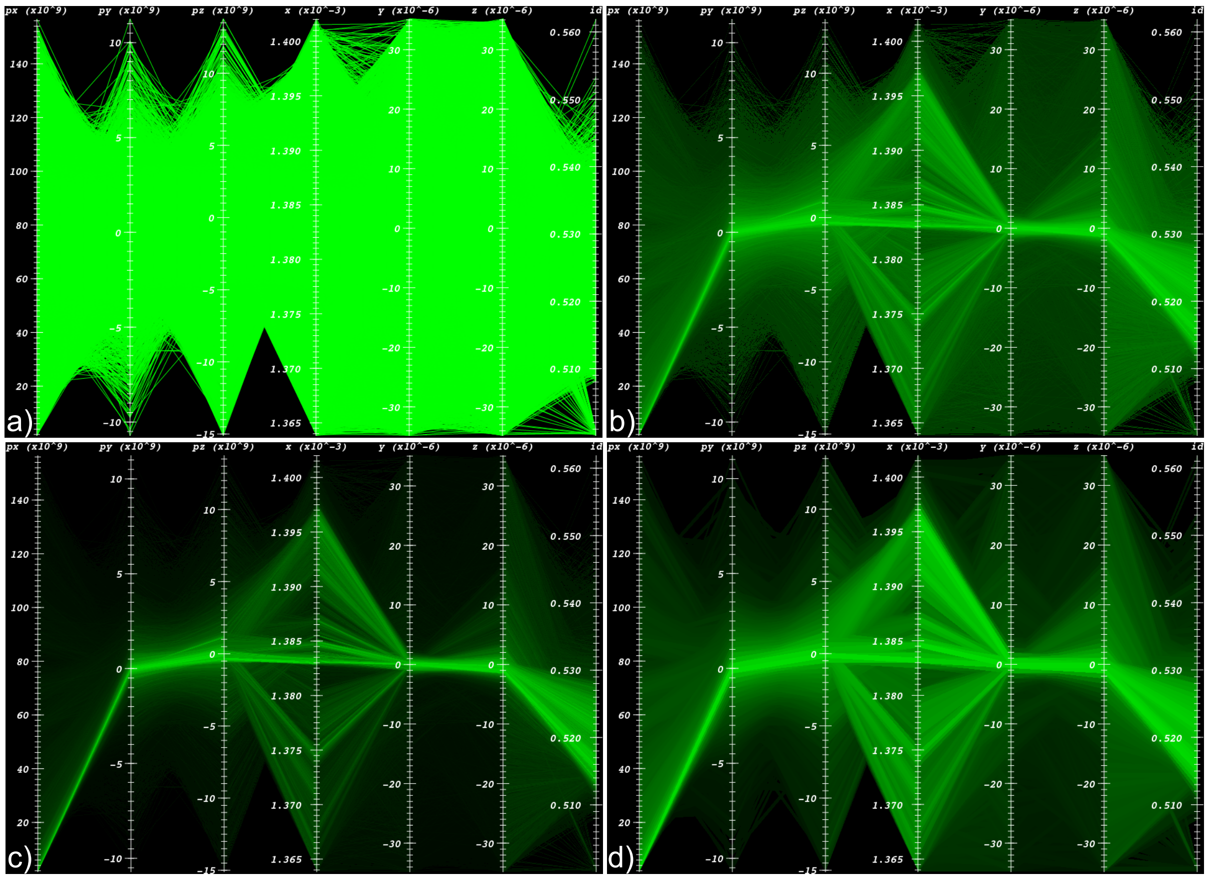


Figure 5.5: Comparison of different parallel coordinate renderings of a subset of a 3D laser wakefield particle acceleration dataset consisting of 256,463 data records and 7 data dimensions. a) Traditional line based parallel coordinates. b) High-resolution, histogram-based parallel coordinates with 700 bins per data dimension. c) Same as in *b*, but using a lower gamma value *g* defining the basic brightness of bins. d) Same rendering as in *b* but using only 80 bins per data dimension. When comparing *a* and *b*, one can see that the histogram-based rendering reveals many more details when dealing with a large number of data records. As illustrated in *c*, by lowering gamma the user can then reduce the brightness of the plot and even remove sparse bins, thereby producing a plot that focuses on the main, dense features of the data. By varying the number of bins the user can then create renderings at different levels of detail.

In practice, parallel coordinates have disadvantages when applied to very large datasets. First, each data record is represented with a single polyline that connects each of the parallel coordinates axes. As data size increases, the plot becomes more cluttered and difficult to interpret. Also, data records drawn later will occlude information provided by data records drawn earlier. Even worse, polyline-based parallel coordinates have a computational and rendering complexity that is proportional to the size of the dataset. As data sizes grow ever larger, these problems become intractable.

To address these problems, this work employs an efficient rendering technique based on two-dimensional (2D) histograms. Rather than viewing the parallel coordinates plot as a collection of polylines, one per data record, VisIt approaches the rendering by considering instead the relationships of all data records between pairs of parallel coordinate axes. That relationship can be discretized as a 2D histogram and then later rendered. This idea was introduced in earlier work by Novotný and Hauser [26]. As illustrated in Figure 5.6, VisIt creates a parallel coordinates representation based on 2D histograms by drawing one quadrilateral per non-empty bin, where each quadrilateral connects two data ranges between neighboring axes. As illustrated in Figure 5.5(a,b), histogram-based rendering overcomes the limitations of polyline-based rendering and reveals much more data detail when dealing with a large number of data records.

In the following, Sections 5.4.2.1 and 5.4.2.2 first describe how to quickly compute 2D histograms (see Section 5.4.2.1) and how to use them to efficiently render parallel coordinates (see Section 5.4.2.2). Having introduced the principle of histogram-based parallel coordinates, Section 5.4.2.3 then compares the use of regular (equal width) and adaptive (equal weight) binned 2D histograms in the context of rendering parallel coordinate views. Afterwards, VisIt's graphical user interface (GUI) of parallel coordinates is described in Section 5.4.2.4.

5.4.2.1 Computing Histograms

In this work, the computation of 2D histograms is implemented at the data I/O stage of VisIt: 2D histograms are computed directly in the file reader, which leverages FastBit for index/query operations as well as histogram computation. This approach has several major benefits within the context of very large, high performance visual data analysis. First, instead of having to read the entire data set and transfer it to the plot to create a rendered image, the internal data transfer and processing is limited to a set of 2D histograms, which are very small when compared to the size of the source data. Second, data I/O is limited to those portions needed for computing the current 2D histogram, i.e., information on the relevant particles in two data dimensions. After computing a 2D histogram, the system can then discard the data the loader may have read to compute the histogram, thus decreasing the memory footprint. An added advantage is the fact that all the computationally and I/O intensive work is done at the beginning of the parallel execution pipeline rather then at the end in the plot. Third, having the histogram computation be part of the file reader allows VisIt to perform such computation at the same stage of processing as parallel I/O, which is one of the most expensive operations in visual data analysis. This approach provides the ability to achieve excellent parallel performance, as described later in Section 5.4.5.

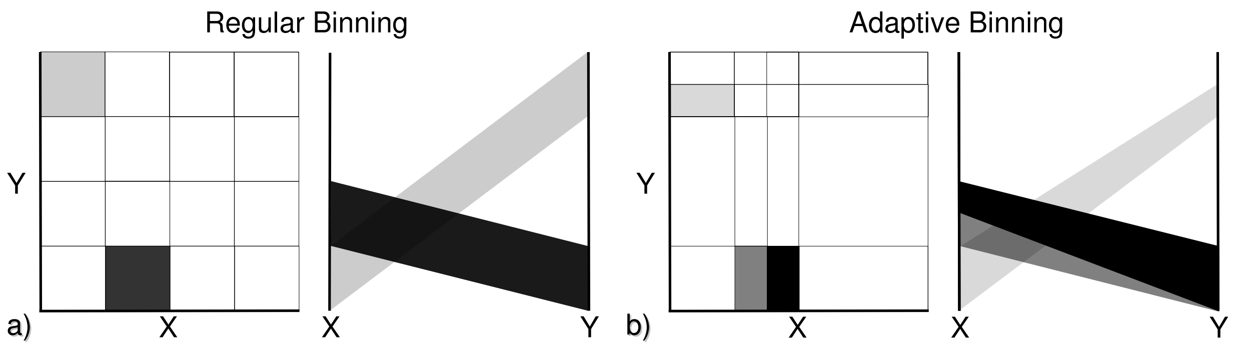


Figure 5.6: Illustration showing the rendering of parallel coordinates based on regularly (a) and adaptively binned histograms (b). By using higher resolution in areas of extremely high density, an adaptive binning is able to represent the general data trends much more accurately.

5.4.2.2 Using Histograms to Render Parallel Coordinates Plots

When using 2D histograms for rendering parallel coordinates plots, the renderer has access to additional information that is not present when using traditional polyline-based methods. In particular, the system knows prior to the rendering the number of records contributing to specific variable ranges between parallel axes. This extra information allows the optimization of various aspects of the data visualization to convey more information to the user. VisIt uses brightness, for example, to reflect the number of records per bin, which leads to improved visual presentation. Assuming that denser regions are more important than sparse regions, VisIt renders bins in back-to-front order with respect to the number of records per bin $h(i,j)$. Figure 5.5(a,b) shows a direct comparison of the same data once rendered using traditional polyline-based parallel coordinates and once using the proposed histogram-based rendering approach.

In order to further improve the rendering, the user can define a gamma value g defining the overall brightness of the plot. As illustrated in Figure 5.5c, a lower g value reduces the brightness of the plot, or even remove sparse bins from the rendering, thereby producing a much less cluttered visualization that focuses attention on the main, dense data features. Since the proposed method is not constrained by a fixed histogram bin resolution, VisIt can easily recompute histograms at higher resolution, or, using adaptive binning, to produce visualizations at varying levels of detail. This feature is important for providing smooth drill-down into finer levels of detail in very large datasets and represents one of the major improvements of VisIt's approach over previous work in which histograms are pre-computed. Figure 5.5d, shows as an example the same rendering as in Figure 5.5b, but using just 80 bins per data dimension. Another example is also provided later in Figure 5.7.

Previous histogram-based parallel coordinates work used histogram-based rendering for the context view and traditional line-based rendering for the focus view. One limitation of this approach is that the focus view may consist of a very large number of data records. VisIt overcomes this limitation by using a histogram-based approach for both the context and focus views. This approach is feasible given the rapid rate at which FastBit can recompute new conditional 2D histograms as explained in Section 5.4.5. The focus view is rendered on top of the context view using a different color to make the focus more easily distinguishable. This approach has the further advantage that it allows VisIt to render both, the focus and the context, at different levels of detail simply by specifying the number of bins per variable. Typically a user specifies lower-resolution histograms for the context view and very high-resolution histograms for the focus view, thereby supporting a high quality and smooth visual drill-down into the focus view.

Similar to rendering the focus on top of the context, VisIt can also create a rendering of multiple timesteps in one parallel coordinates plot. To do so, the user assigns a unique color to each timestep and the system then renders the individual plots — each representing one timestep — on top of one other. As described later in Section 5.8.1, such a rendering can be helpful to identify the general temporal changes in the data. In practice, temporal parallel coordinates are most useful when analyzing some characteristic subset of the data.

5.4.2.3 Adaptive and Uniform Histogram Binning

Regular histograms, with uniform, equal-sized bins, are well suited for high resolution renderings but have substantial disadvantages when creating low level-of-detail views where the number of bins per variable is much smaller than the number of pixels per parallel axis. In such a case each bin may cover regions of varying data density, e.g., large areas of a bin may contain only few data records while other areas contain many. This work describes another improvement over previous work, namely FastBit can compute and VisIt render with adaptive, rather than uniform histogram bins. With adaptive binning, each bin of the histogram contains approximately the same number of data records, which may offer advantages in certain situations [146].

As an example, Figure 5.7 shows a comparison of data visualized using $32x32$ uniform versus adaptive histogram bins. In comparison to a uniform binning, adaptive binning discards some features in sparse areas of the data to preserve more information in dense areas. Adaptively binned histograms may ease comparison of selections with general data trends. As illustrated in Figure 5.6, when using adaptively binned histograms, a more generalized rendering is required to allow rectangles to connect different sized ranges on neighboring axes. Also, since the area $a(i,j)$ covered by each bin is no longer constant, VisIt needs to compute the brightness of each bin, and then assign rendering order based on the actual data density per bin $p(i,j) = \frac{h(i,j)}{a(i,j)}$, rather

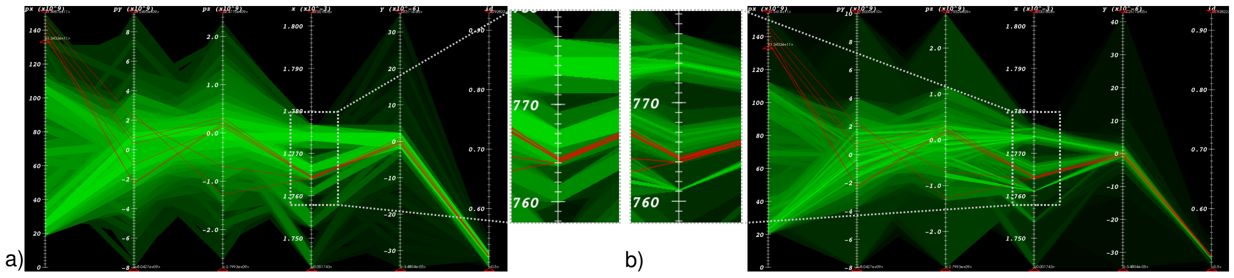


Figure 5.7: Histogram-based parallel coordinates using $32x32$ regularly binned histograms (a) and adaptively histograms (b). Compared to regular binning, the adaptive binning preserves more details in dense areas while discarding some details in sparse areas of the data. An adaptively binned plot may ease comparison of selections (red) with structures present in the data in low level of detail views.

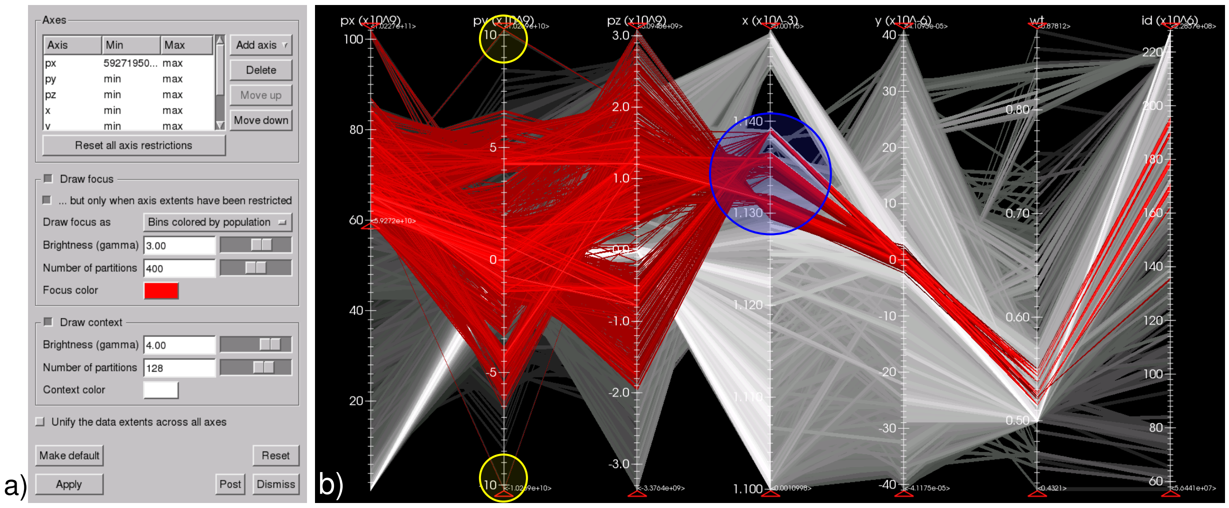
than based on $h(i, j)$ directly. Uniformly binned histograms are in general very well suited for high resolution renderings of the data while adaptively binned histograms may be advantageous for low resolution renderings.

In other applications, such as statistical analysis, one is often interested in both the main data trends as well as outlier behavior. In order to achieve an optimal low level-of-detail rendering for such applications, one could further restrict the minimal density p of bins during computation of the adaptive binning to ensure that details in sparse areas of the data are represented accurately. As proposed by Novotný and Hauser [26] one may also employ a separate outlier detection scheme for this purpose in which data records located in bins of extremely low density p are rendered as individual lines resulting in a hybrid approach of line-based and histogram-based parallel coordinates.

5.4.2.4 Parallel Coordinates in VisIt

In VisIt, parallel coordinates are implemented as a *plot* plugin. The user interface of parallel coordinates consists of three main parts: i) the attributes window, ii) the *Viewer* displaying the actual plot, and iii) the *Extends* tool.

The attributes window (see Figure 5.8 a), is used to specify the many different settings of parallel coordinates. The user can here specify different visualization options, such as, the rendering resolution, as well as the data variables to be displayed. For the focus view, three different rendering options are available: i) lines, i.e., render the focus using line-based parallel coordinates, ii) constant color, i.e., use a histogram-based rendering with constant color, and iii) varying color, i.e., use a histogram-based rendering and vary brightness of color based on density. In the context of FastBit enabled VisIt, all histograms required for rendering of parallel coordinates are



Axes

Axis	Min	Max
px	59271950...	max
py	min	max
pz	min	max
x	min	max
y	min	max

Draw focus

... but only when axis extents have been restricted

Draw focus as: Bins colored by population

Brightness (gamma): 3.00

Number of partitions: 400

Focus color: 🟥

Draw context

Brightness (gamma): 4.00

Number of partitions: 128

Context color: ⬜

Unify the data extents across all axes

Figure 5.8: Parallel coordinates in VisIt: a) GUI for defining the different parameters of parallel coordinates, such as, the resolution (*Number of partitions*), color, or brightness (*gamma*) of the context and focus histograms. b) Parallel coordinates with context particles (gray) and focus (red). The red, triangle-shaped sliders are used to define the selection query $px > 5.9272{*}10^{10}$ shown in the focus (red). The selection defines three distinct groups of particles along the x axis (blue circle). In py several data lines diverge from the main data trend of the selection indicating that the according particles are outliers (yellow circle).

computed in the file reader using FastBit, i.e, no raw data needs to be transferred to the plot but only a set of much smaller histograms.

The VisIt Viewer then defines the standard interaction mechanisms for execution of, e.g., zooming and panning of the view. Here the user can also enable the Extends tool.

The Extends tool is used to perform selection operations in parallel coordinates and, hence, to define the focus view. The tool consists of two sliders rendered on top of each parallel axis (see Figure 5.8 b). By moving the sliders using the mouse a user can intuitively define a minimum and maximum value in each displayed data dimension. Together, the value ranges define a multivariate range query describing the data of interest. Again, the according conditional histograms required to render the focus are directly computed using FastBit. This means not only that no raw data needs to be transferred to the plot, but more importantly, that VisIt only needs to access the data associated with the selected data records allowing it to quickly recompute the focus histograms.

As illustrated in Figure 5.8 b, when performing data selection in parallel coordinates a user receives immediate feedback about important properties of the current selection. Outliers, e.g., appear as single data lines diverging from the main data trend(s) (yellow circles in Figure 5.8 b).

The appearance of several distinct groups of data lines may indicate that the selection crosses different cluster suggesting that one may want to refine the selection accordingly (blue circle in Figure 5.8 b). Finally, the user may also compare the data trend(s) shown in the focus with the data displayed in the context.

5.4.3 Data Selection

In the proposed system, there is synergy between presentation of information to the user and specification of queries. The parallel coordinates plot serves to present context and focus data views to the user, and also serves as the mechanism for specifying a multivariate Boolean range query. In a typical use case (see, e.g. Section 5.8.1), a multivariate range query might take the form of $px > 10^9$ && $py < 10^8$ && $y > 0$, which selects high momentum particles in the upper half of the beam. Similar conditions can be formulated to define arbitrary subsets of particles with interesting momentum and spatial characteristics.

Besides parallel coordinates, the user can also define multivariate range queries via other VisIt operators. Using, e.g., a *Threshold* operator, a user can quickly generate subset plots of the data without having to go through a manual, step-by-step selection process. In case the user has a priori knowledge of the data subset of interest, a Threshold operator may, therefore, be more efficient. In contrast to the Threshold operator, parallel coordinates do provide important feedback in the selection process. Parallel coordinates are, therefore, the preferred interface for defining data selections in cases where the boundaries of the data subset of interest are unknown.

Once the user specifies a multivariate range condition, those conditions are passed back upstream in the system to the FastBit-enhanced HDF5 reader for processing, either to compute new histograms or to extract data subsets that match the query for downstream processing. In the case of data subsetting, FastBit locates those data records that satisfy the query and then pass them along for downstream processing. For conditional histograms, FastBit computes new histograms using the query conditions as well as a histogram specification: the number of bins and the bin boundaries.

Once an interesting subset has been identified, yet another form of query can be issued in order to identify the same data subset at different points in time. This type of query is of the form $ID \in (id1, id2, ..idn)$, where there are n particles in the subset. Again, such queries can be processed efficiently by FastBit and only the relevant set of particles is extracted and then passed along to visual data analysis machinery. This processing step offers a huge performance advantage: a technique without access to the index information must search the entire dataset for particle identifier matches. In VisIt ID-based selection is implemented via the novel concept of Named Selections (see Section 5.4.3.1). By issuing an identifier query across the entire (or parts of the) time sequence, VisIt can construct particle paths, which reveal valuable information about how particles are accelerated over time (see Section 5.4.3.2).

5.4.3.1 Named Selection

Named Selections are a new concept in VisIt allowing selection of data subsets based on ID, here particle IDs defined as a separate data variable. The concept of ID-based selection allows the same particle subset to be extracted from different timesteps and in this way defines the foundation for tracing of particle subsets over time. In contrast to an operator, plot, or database plugin, Named Selections define a new concept of the core VisIt system.

To create a Named Selection, a user issues an according request by calling the *CreateNamedSelection(...)* function via VisIt's CLI interface or by using an according macro[8]. This request is communicated to the VisIt compute engine via a corresponding remote procedure call (RPC) (see Section 2.4.1). In this call the client sends a user specified name for the Named Selection and the Id of the plot the Named Selection should be created from to the engine running on the server. The engine then creates a new Named Selection that selects all data objects displayed by the input plot (i.e., the IDs of the according data objects, here particles). All components of the according visualization pipeline that influence the selection (e.g., operators or plots that modify the set of displayed particles), therefore, need to overwrite according virtual functions to enable the engine to create a Named Selection based on the modifications they define. For example, a Threshold operator may modify the list of displayed data objects by removing all parts of the data that do not suffice a given set of threshold criteria (e.g., select only particles with $px > 1e10$). Similarly, a parallel coordinates plot may modify a Named Selection based on the selection criteria defined in the focus of the plot.

Besides CreateNamedSelection, VisIt supports the following additional RPC's specific to Named Selections: i) *ApplyNamedSelection(...)*, i.e., apply the selection with the given name to a plot, ii) *SaveNamedSelection(...)*, i.e., save a Named Selection to disk, and iii) *LoadNamedSelection(...)*, i.e., load a previously saved Named Selection from disk.

Named selections are a general concept, i.e., they can be used in combination with all datatypes that allow for ID-based selection. In context of FastBit-enabled VisIt, ID-based selections defined by Named Selections are evaluated directly by FastBit in the filereader. This

[8] *Macros* are used in general to represent a set of instructions in an abbreviated format. In VisIt a macro consists of a Python script, a function name, and a user defined name. Macros can be represented in the VisIt GUI as a single button providing convenient user-access to macro functions.

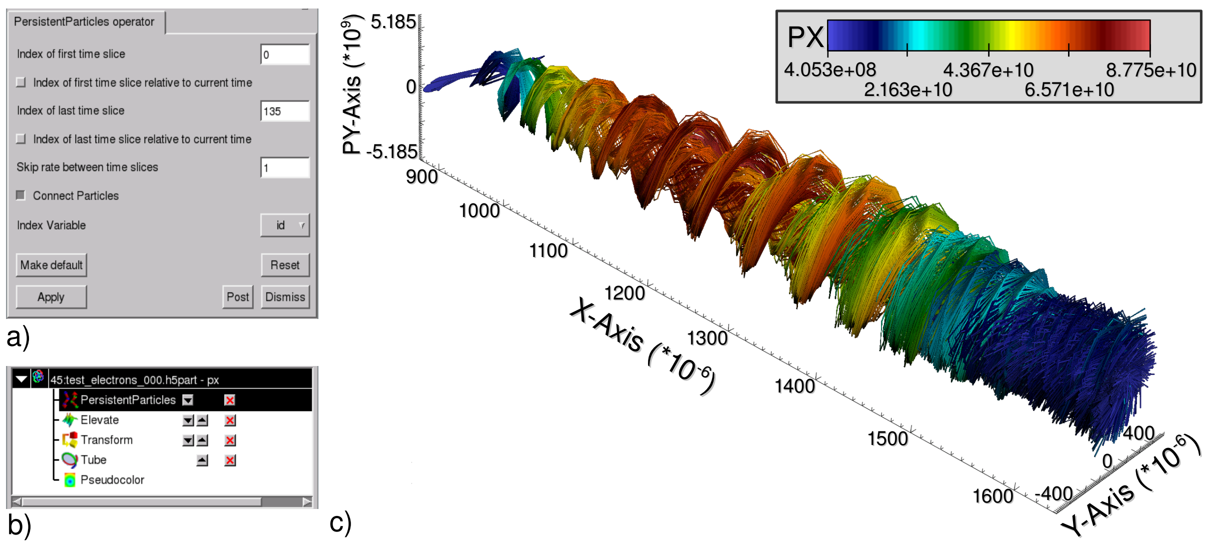
means, when a NamedSelection is applied then the filereader loads only the data associated with the selected data objects (here particles), greatly reducing the workload of the then following operations. In case of database plugins that do not support ID-based selection, VisIt performs the according filtering operation itself after the data has been loaded.

5.4.3.2 Particle Tracing and Particle Path Visualization

Using the concept of Named Selections a user can define a beam and trace the beam particles over time. Being able to follow a set of particles step-by-step through time is very informative because it allows scientists to investigate where particles selected at timestep *a* are located at timestep *b*. This approach furthermore allows for identification of different substructures of a beam. While the selected particles may form a compact bunch at timestep *a* they may define different clusters in timestep *b*. However, when focusing on a single timestep at a time, large parts of the temporal context are lost. Particle path visualizations overcome this limitation by showing the complete (or parts) of the path a particle is moving along over time.

In VisIt particle paths are computed using the so called *Persistent Particles* (PP) operator. The Persistent Particles operator iterates through the time series and connects the location of corresponding particles via line segments. In most cases the Persistent Particles are combined with a Named Selection, i.e., the user usually traces only a small subset of particles over time (here, e.g., a particle beam). Using the concept of Named Selections has the advantage that the Persistent Particles operator does not need to know whether all or only a subset of particles should be traced. Furthermore, VisIt can evaluate Named Selections fast using FastBit so that only the data of the selected particles is passed on to the Persistent Particles operator greatly improving the performance of particle tracing in VisIt. When computing particle paths, one ID-query is issued for each relevant timestep, e.g., when tracing particles from timestep $t = 20$ to $t = 29$ the system needs to evaluate ten ID queries. Using the proposed framework the particle paths for a single beam are computed within seconds compared to hours using the original scripts (see Section 5.4.5 for details). VisIt, hence, enables scientist to use particle tracing as a regular tool in the analysis process greatly improving efficiency and accuracy of the overall analysis process.

Figure 5.9 a, shows the GUI of the Persistent Particles operator. The user can here specify: i) the start and end time for the tracing, ii) a skip rate defining whether every (*skip rate=1*), or, e.g., only every other timestep (*skip rate=2*) should be considered, iii) whether particles should be connected to define paths, and iv) which variable should be used to identify corresponding particles (here *ID*). The *Connect Particles* option defines whether the data from the different timesteps should simply be merged (*Connect=false*), or whether particle paths should be created


PersistentParticles operator

Index of first time slice: 0

☐ Index of first time slice relative to current time

Index of last time slice: 135

☐ Index of last time slice relative to current time

Skip rate between time slices: 1

▨ Connect Particles

Index Variable: id

Make default | Reset

Apply | Post | Dismiss

45:test_electrons_000.h5part - px
- PersistentParticles
- Elevate
- Transform
- Tube
- Pseudocolor

Figure 5.9: a) GUI of VisIt's *Persistent Particles* operator used for tracing of particles over time. b) Example VisIt pipeline for particle tracing. After computing the geometry of particle paths using *Persistent Particles* (settings shown in *a*) py is mapped to the z dimension of the plot using an *Elevate* operator. The paths are then scaled (*Transform* operator) and transformed to tubes (*Tube* operator) to improve the visual appearance of the plot. Particle paths are finally rendered using a *Pseudocolor* plot where color represents px (see legend). c) Particle path visualization of a particle bunch in a 2D particle dataset created using the pipeline shown in *b*.

by connecting corresponding particle locations with line segments (*Connect=true*). The start and end time for the tracing can be defined either as absolute times (i.e, trace particles from timestep a to timestep b) or relative to the current time. Relative start and end times allow a user, e.g., to specify that VisIt should trace particles n timesteps backward in time ($start = -n$) and m timesteps forward in time ($stop = +m$). This concept enables the user to create visualizations and animations in which particle paths change dynamically with time.

Figure 5.9 b, shows an example pipeline for visualization of particle paths in VisIt. The resulting visualization is shown in Figure 5.9 c. In VisIt, particle paths — computed by the Persistent Particles operator — are visualized using a *Pseudocolor* plot (see Figure 5.9 b, c). The Pseudocolor plot allows the user to color particle paths according to a selected variable and, e.g., also change the opacity of particle paths. The Persistent Particles operator computes particle paths in physical space (i.e., (x,y) in 2D and (x,y,z) in 3D). In the case of 2D particle paths an *Elevate* operator may be used to extend the paths to the third dimensions by mapping the values of a selected variable to z (see Figure 5.9 b, c). To improve the visual appearance of particle paths a user may then apply a *Tube* operator to display paths as three-dimensional tubes rather than lines. Finally, a user may apply additional geometric transformations using a

Transform operator to, e.g., scale particle paths or display paths in spherical instead of Cartesian coordinates. Figure 5.9 c, shows an example particle path visualization in (x, y, py) space with paths being colored according to px.

5.4.4 Visual Exploration of Laser Wakefield Accelerator Simulations

This section describes the general model of a typical visual analysis of laser wakefield particle acceleration data. A detailed example use case study based on the described model is presented later in Section 5.8.1. The introduced use model, while applied here to accelerator science, is generally applicable to a broad set of applications.

Figure 5.10 illustrates the general workflow of a typical visual analysis. Initially, a user defines a selection of particles at a single timestep. Selection of particles is performed in an iterative process in which the user first defines a set of different thresholds using parallel coordinates (see Figure 5.10 a, left). Validation of the selection is initially usually done using the parallel coordinates plots itself. Having identified a particle subset of interest the user validates and analyzes the selection using different types of plots (e.g., a Pseudocolor plot) (see Figure 5.10 a, right). Depending on the insight the user has gathered from the different visualizations, the user may then return to the parallel coordinates plot to further improve the selection. This process of iterative selection and validation supports accurate selection of particle subsets of interest.

Having defined a particle subset of interest the user then typically investigates the temporal behavior of the selected particles. This is done either directly via inspection of the paths of the selected particles (see Figure 5.10 b) or by tracing the individual particles through time by changing the timestep (see Figure 5.10 c). Figure 5.10 b, shows the paths of the particles selected in the parallel coordinates plot shown in Figure 5.10 a. Based on the paths different sets of particle injection occurring at four different times can readily be identified.

The identified particle subgroups are not only injected at different times but also define distinct clusters in physical space during early timesteps. As the next step in a typical analysis the user, therefore, modifies the current timestep, i.e., the user traces the particles through time. In the context of laser wakefield accelerator data, particles are typically selected at a late timesteps and are then traced backward in time. To separate the different groups of particles behaving differently over time, the user refines the selection based on information from a different timestep. In the example shown in Figure 5.10 c, the user removed the two sets of particles that were injected early from the selection by applying an additional threshold in x at an early timestep. As in the initial selection process, a user may iteratively refine the selection at that timestep based on insight from the visual validation.

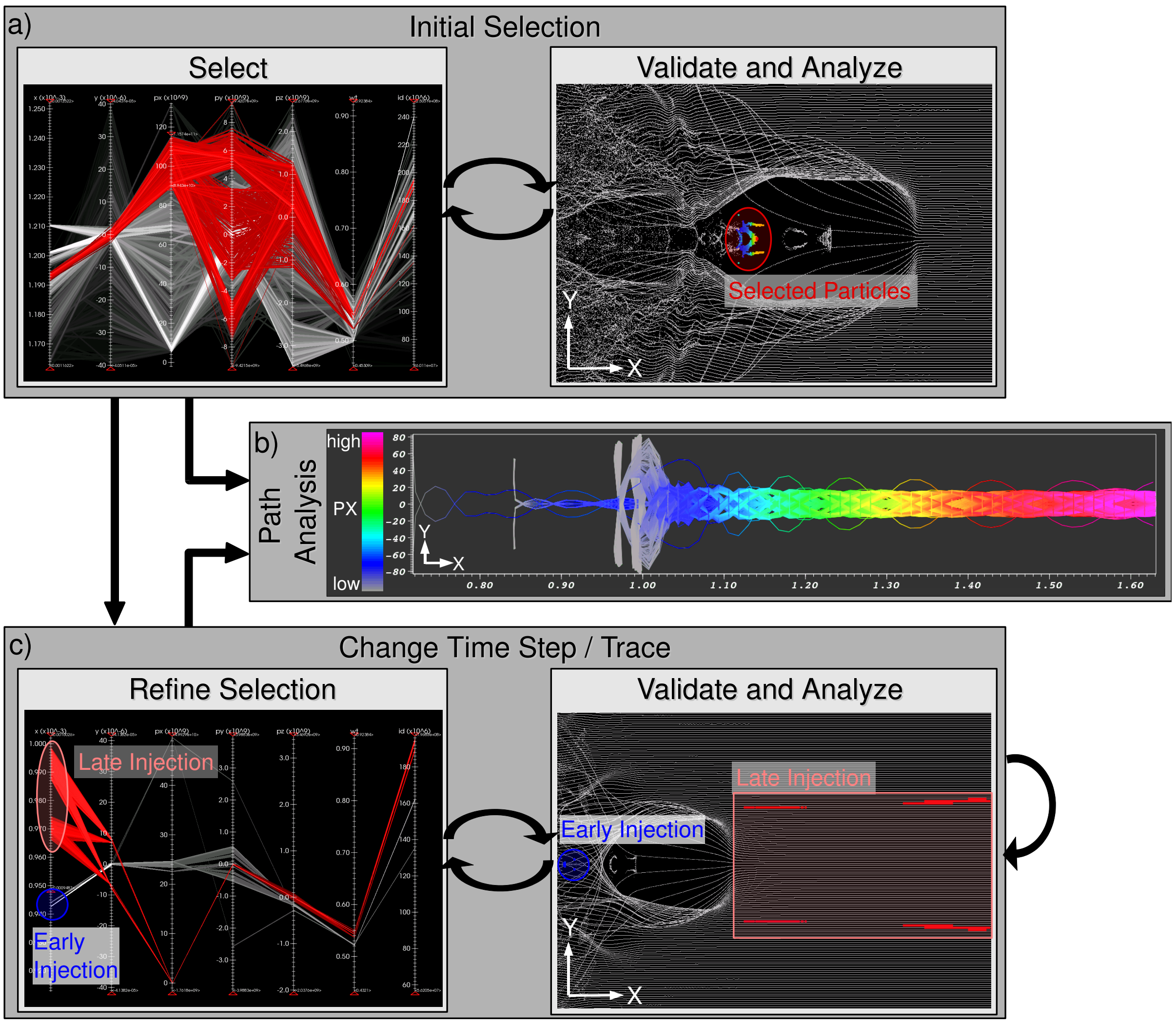


Figure 5.10: Use model for the visual exploration of particle data from laser wakefield accelerator simulations. a) Initially a user selects a particle bunch of interest using parallel coordinates. Various other plots (e.g., scatter-plot or Pseudocolor plot) serve then as means for validation and analysis of a particle selection. Having defined a selection, a user commonly computes the temporal paths of the selected particles (b) or modifies the timestep shown (c), to investigate the behavior of particles over time. In this process a user may identify different subsets of particles that behave differently over time and refine the selection accordingly. Figure *a* and *c* each show a parallel coordinates plots (left) and an (x,y) point plot (right). Particles that are part of the selection are shown in color whereas other particles are shown in gray. The particle traces shown in *b* are based on the selection defined in *a*.

Afterwards, a user may then again, either analyze the particle paths of the refined selection or trace the particles through time by modifying the current timestep. In this process a user may identify various other substructures of the initial selection and further refine the selection based on information from different timesteps.

The concept of iterative, interactive refinement and validation of data selections enables accurate definition of data features. In particular compared to an analysis based on animations of static plots and non interactive scripts for data selection — a methodology commonly used in practice — the proposed methodology enables a more detailed and accurate data analysis and enables efficient analysis of even extremely large simulation datasets which is prohibitively slow otherwise.

5.4.5 Performance Evaluation

This section presents results of a study aimed at characterizing the performance of the described implementation under varying conditions using standalone, benchmark applications. These unit tests reflect the different stages of processing presented earlier in Section 5.4.4 (see also Section 5.8.1). First, the presented study examines the serial performance of histogram computation in Section 5.4.5.1: histograms serve as the basis for visually presenting data to a user via a parallel coordinates plot. Second, Section 5.4.5.2 examines the serial performance of particle selection across all timesteps of simulation data. Finally, Section 5.4.5.3 investigates the parallel scalability characteristics of the histogram computation and particle tracking implementations on a Cray XT4 system.

The serial performance tests in Sections 5.4.5.1 and 5.4.5.2, use a 3D particle dataset consisting of: 30 timesteps worth of accelerator simulation data, each timestep has about 90 million particles and is about $5GB$ in size (plus $\approx 2GB$ for the index). The aggregate dataset size is about $210GB$, including the index data. The system stores and retrieves simulation and index data using HDF5 and a veneer library called *HDF5-FastQuery* [122]. HDF5-FastQuery presents an implementation-neutral API for performing queries and obtaining histograms of data. The serial performance tests were conducted on a workstation equipped with a 2.2Ghz AMD Opteron CPU, 4GB of RAM and running the SuSE Linux distribution.

The serial performance tests in Sections 5.4.5.1 and 5.4.5.2 measure the execution time of two different implementations that are standalone applications created for the purpose of this performance experiment. One application, labeled *FastBit* in the charts, uses FastBit for index/query and histogram computation. The other, labeled *Custom* in the charts, does not use any indexing structure, and therefore performs a sequential scan of the dataset when computing histograms and particle selections. Note, in order to enable a fair comparison this study uses a dedicated Custom code which shows better performance then the IDL scripts currently used by the science collaborators of the LOASIS program who supported this work.

5.4.5.1 Computing 2D Histograms

This section describes a set of tests aimed at differentiating performance characteristics of computing both unconditional and conditional histograms. The unconditional histogram is simply a histogram of an entire dataset using a set of application-defined bin boundaries. A conditional histogram is one computed from a subset of data records that match an external condition.

Unconditional Histograms

In the use model of laser wakefield accelerator analysis, the computation of the unconditional histogram is a "one-time" operation. It provides the initial context view of a dataset to the user.

For this test, the test program varies the number of bins in a 2D histogram over the following bin resolutions: 32x32, 64x64, 128x128, 256x256, 512x512, 1024x1024, and 2048x2048 bins. All histograms span the same range of data values: increasing the bin count results in bins of finer resolution. The results of this test are shown in Figure 5.11. Since both the FastBit and Custom applications need to look at all the data records to compute an unconditional histogram, large variations in performance with changing number of bins are not expected. FastBit is generally faster than the Custom code throughout primarily because of the difference in organization of the histogram bin counts array. FastBit uses a single array, which results in a more favorable memory access pattern. FastBit computes adaptive histograms by first computing a higher resolution regularly binned histogram and then merging bins. The computation time for adaptively binned histograms shows only a a minor, constant increase compared to the case of regular binning because merging bins is a fairly inexpensive process and increasing the number of bins has no significant effect on the performance of regularly binned histograms.

Conditional Histograms

In contrast to the unconditional histogram, which is a one-time computation, the process of visual data exploration and mining relies on repeated computations of conditional histograms. Therefore, this work focuses on achieving good performance of this operation to support interactive visual data analysis. This set of tests are based on the described use model; a change in the set of conditions results in the examination of a greater or lesser number of data records to compute a conditional histogram. This set of conditions reflects the repeated refinement associated with interactive, visual data analysis.

This test is based on parameterizing the number of hits resulting from range queries of the form $px > \ldots$ defined as the histogram conditions. With increasing thresholds of particle momen-

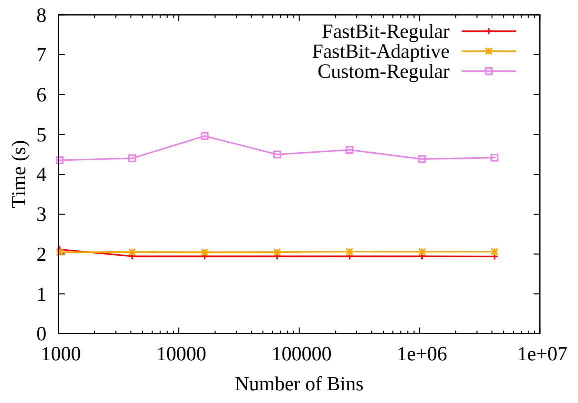


Figure 5.11: Timings for serial computation of unconditional histograms.

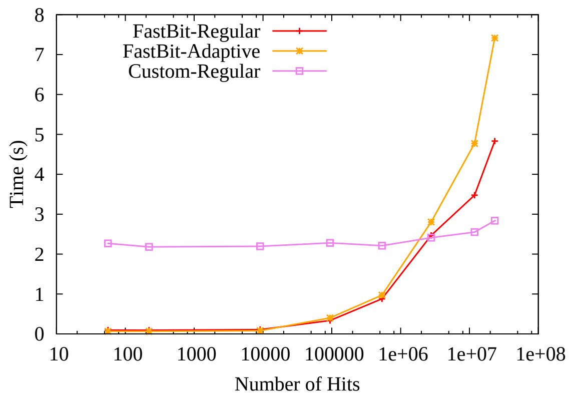


Figure 5.12: Timings for serial computation of conditional histograms.

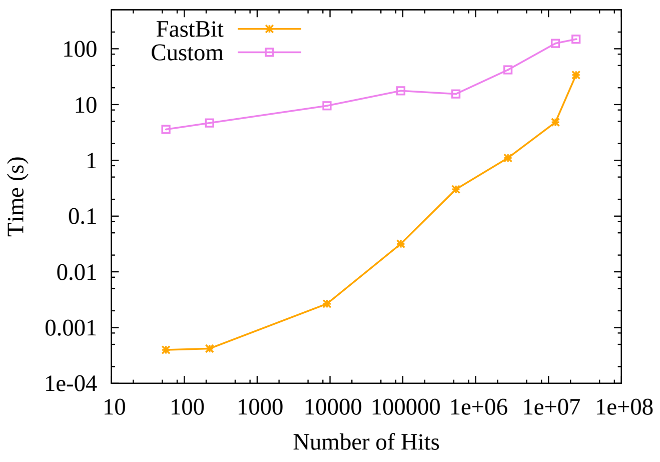


Figure 5.13: Timings for serial processing of particle identifier queries.

tum in the *x* direction (*px*), fewer data records (particles) match the condition and contribute to the resulting histogram. In these tests, the number of bins is defined to be constant at 1024x1024.

Figure 5.12 presents results for computing conditional histograms using FastBit and the Custom application. For small number of hits, the FastBit execution times are dramatically faster than the Custom code that examines all data records. Also, in this regime the regular and adaptive binning show similar performance. While for unconditional histograms the minimum and maximum values are known for each variable, FastBit here needs to compute these values from the selected data parts in order to compute the adaptive binning. Due to this fact the performance of the adaptive binning decreases compared to regular binning for very large selections. It is important to note that visual analysis queries typically isolate a small number of particles – from tens to thousands – and FastBit provides outstanding performance in this regime. As the number of hits increase, approaching 1M and 10M+ particles – which is a significant fraction of the 90M total particles – a sequential scan through all the data records produces better results. This performance change is due to the fact that FastBit computes the histogram in two separate steps. It first evaluates the user-specified conditions to select the appropriate values, and then counts the number of values in each bin of the histogram. The selected values are passed from the first to the second step as an intermediate array with as many elements as the number of hits. It is expensive to pass this intermediate array through memory when it is large. Since the intended application primarily have a small number of hits, using FastBit is more efficient. The adaptive binning strategy requires an additional pass through the data to determine the actual minimum and maximum values, which demands significant amount of time.

5.4.5.2 Particle Selections

Following the described use model, once a user has determined a set of interesting data conditions, like particles having a momentum exceeding a given threshold, the next activity is to extract those particles from the large dataset for subsequent analysis. This set of tests aims to show performance of the particle subsetting part of the processing pipeline.

The execution time for this task is clearly proportional to the size of the selection – the time required to find a set of particles in a large, time-varying dataset varies as a function of the size of the particle search set as well as the size of the simulation data itself. The test parameterizes the search set size by varying the number of particles to search for over values ranging from 10, 100, 1000, ... , up to 20M particles.

The Custom code uses a sequential scan of the entire dataset to search for particles in the search set. For each data record, it compares the particle ID of the record to the search set using an efficient algorithm: if the size of the search set is S, then the search time is $log(S)$. If there are N data records in the entire dataset, the computational complexity of the entire algorithm is $Nlog(S)$. In contrast, the worst-case time required to locate a set of identifiers are expected to be proportional to the number records found [53].

Figure 5.13 presents results for running ID queries using FastBit and the Custom code for one timestep. For relatively small number of identifiers, FastBit is about four orders of magnitude ($10^4\times$) faster than the Custom code. As the number of identifiers involved increases, the relative difference becomes smaller. When 20,000,000 of identifies are involved, FastBit is still three times faster.

5.4.5.3 Scalability Tests

Whereas the previous sections focused on the serial performance of computing histograms and performing particle subset selections, this section focuses on the scalability characteristics of both algorithms.

The platform for these tests is `franklin.nersc.gov`[9], a 9,660 node, 19K core Cray XT4 system. Each of the nodes consists of a 2.6GHz, dual-core AMD Opteron processor[10] and has 4GB of memory and runs the Compute Node Linux distribution. One optimization the test code uses on this machine at all levels of parallelism is to restrict operations to a single core of each node. This optimization maximizes the amount of memory and I/O bandwidth available to each process in the parallel performance tests. On this platform, the Lustre Parallel Filesystem serves data to each of the nodes. The nodes used in the study are a small fraction of a larger shared facility with a dynamic workload. The scalability tests cover parallelism levels over the following range: 1, 2, 5, 10, 20, 50, and 100 nodes of the Cray XT4.

As in the previous sections, here wall-clock times are reported which encapsulate CPU processing and I/O. Speedup factor are computed as the ratio of time taken by a single node to the times taken by the node subset to complete a task. The here described tests implement a strong scaling, i.e., they keep the problem size fixed at 100 timesteps for all cases.

The dataset used in this study has 100 timesteps; each timestep has 177 million particles and is about $10GB$ in size. The aggregate dataset size is about $1.5TB$, including the index data. The scalability tests employ a fairly simple form of data partitioning: namely they assign subsets of timesteps (corresponding to individual HDF5 files) to individual nodes for processing. The subsets are statically assigned to nodes in a strided fashion.

Following the theme of typical use cases this work reports times for realistic science usage scenarios. For histogram computation, the test code generates five parallel histogram pairs for the position and momentum fields. The histograms consist of 1024x1024 bins, which is a reasonable upper limit given typical screen resolutions. For conditional histograms, the query $px > 7 * 10^{10}$ is used. To evaluate the performance of the particle tracking, the query $px > 10^{11}$ — which results in 500 hits — is used. All of these choices are grounded in discussions with science collaborators from the LOASIS program and reflect reasonable ranges and thresholds.

Figure 5.14 presents results from parallelizing the computation of both conditional and unconditional histograms over multiple nodes. As expected, the computation time decreases as

[9]For more details about Franklin see `http://www.nersc.gov/nusers/systems/franklin/about.php`.

[10]In the meantime NERSC has updated the compute nodes of its supercomputer franklin to AMD Opteron 2.3 GHz Quad Core processors and 8GB memory. The tests described here were executed before this update.

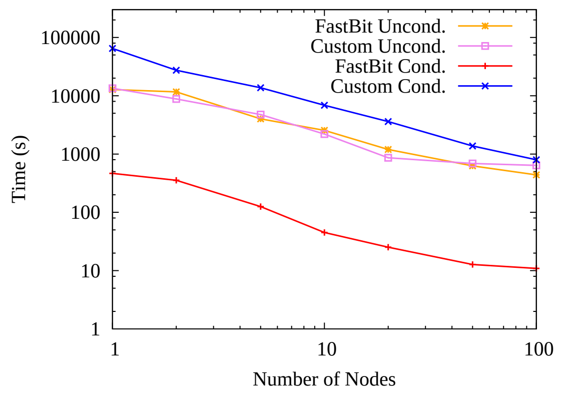


Figure 5.14: Timings for parallel computation of histograms.

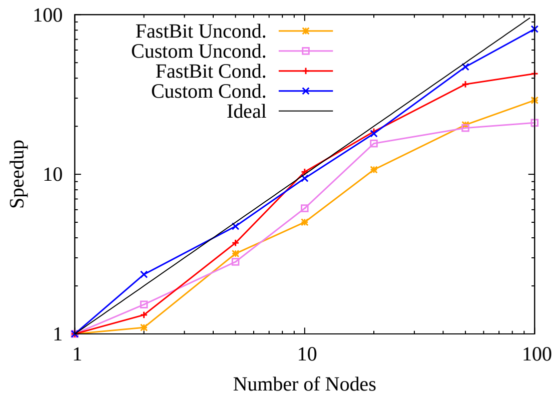


Figure 5.15: Scalability of parallel histogram computation.

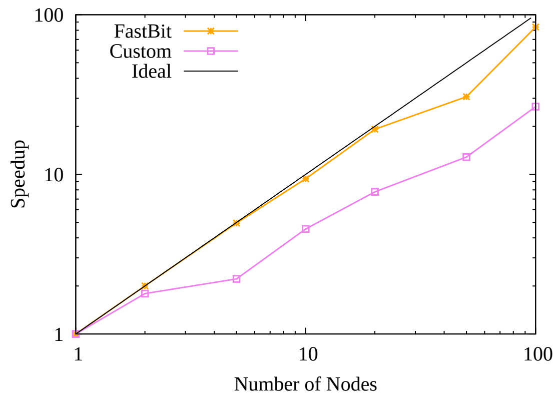


Figure 5.16: Scalability of parallel particle tracking.

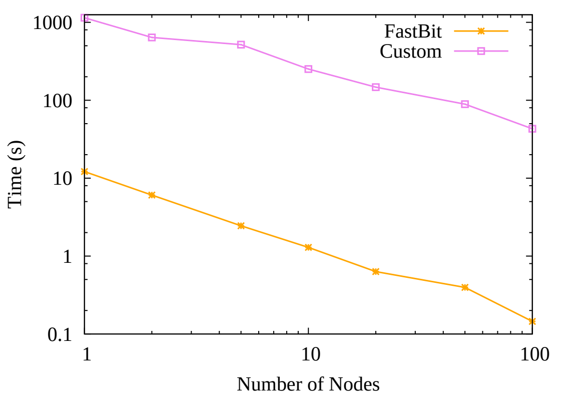


Figure 5.17: Timings for parallel particle tracking.

more nodes are added. Similar to the serial case, there is no major difference between FastBit and the Custom application for computing the unconditional histogram since both implementations examine all data records. For computing the conditional histograms, FastBit maintains its advantage over the Custom application. Figure 5.15 presents the speedup factors corresponding to this computation; showing a very favorable speedup. This is to be expected since the nodes can perform their computations independent of others.

Figure 5.17 presents results from particle tracking over 100 timesteps. Similar to the serial case, FastBit is faster than the Custom application, and maintains its advantage when run in parallel. Figure 5.16 demonstrates that the parallel implementation of the particle tracking shows excellent scalability. When using 100 nodes, FastBit is able to track 500 particles over 1.5TB of data in 0.15 seconds. In contrast, the IDL scripts currently used by the LOASIS researchers who supported this work take $\approx$2.5 hours to track 250 particles for a small 5GB dataset. Furthermore, owing to IDL limitations, the scripts cannot load all required data for extremely large 3D datasets requiring incremental data access which in turn leads to a further decrease in performance. Thus, this research represents a dramatic improvement for the scientist's workflow: what formerly required many hours can now be processed in less than a second.

5.5 Automatic Beam Detection Using Bunch-lifetime Analysis and Fuzzy Clustering

While interactive exploration of the laser wakefield accelerator (LWFA) simulation data using high-performance visual analysis is effective, it is limited with respect to the analysis of large numbers of datasets. The data exploration methods described so far rely on substantial manual input by the user and may, hence, be time-consuming. In order to enable analysis of large databases of simulation data, auxiliary mechanisms are needed to automate the most time-consuming parts of the analysis. One key problem in the analysis of LWFA simulation data is the detection of particle beams.

To address this problem, this section describes *automatic beam detection*, a novel algorithm for automatic classification of the highest-energy particle beam in LWFA simulations. First, the algorithm identifies a set of beam-point candidates describing the location of high-densities of accelerated particles at individual timesteps. Afterwards, the so called bunch lifetime diagram is constructed by organizing and pruning the beam-point candidates as nodes in a minimum spanning tree. Independent of the bunch lifetime analysis, the algorithm partitions the data using fuzzy clustering to detect timesteps that contain a high-energy particle beam. By combining the results from fuzzy clustering and the bunch lifetime analysis, estimation of the highest-energy beam in a LWFA simulation becomes possible.

Algorithms like the automatic beam detection or the automatic beam path analysis described in Section 5.6, promise to support a more focused and efficient analysis process. Automatic analysis algorithms are, hence, one main component to enable the analysis of large collections of LWFA simulations.

In the context of the automatic beam detection approach introduced in this section this work describes the following new contributions:

(i) A new method for identification of features of high particle density indicating the location of potential particle beams in space as well as their lifetime is described (see Section 5.5.2 and 5.5.3).

(ii) The application of fuzzy clustering for detection of potential particle beams at single timesteps is introduced (see Section 5.5.4.

(iii) By combining the results from the bunch lifetime analysis and fuzzy clustering this work enables the estimation of the highest-energy beam formed during the course of a LWFA simulation (see Section 5.5.5).

(iv) Finally, the proposed algorithm is validated using four different simulation datasets illustrating the effectiveness of the proposed approach (see Section 5.5.6).

5.5.1 Design of the Analysis Pipeline

The goal here is to on design an analysis framework to aid physicists in detecting beam formation and characterizing beams. The beams of interest constitute a condensed group of particles exhibiting high momentum along the *x*-axis (px), i.e., high momentum along the laser propagation axis. Furthermore, a high-quality beam is condensed, i.e., it should have small spread in the spatial-energy dimensions (i.e., in 2D (x, y, px, py)). The proposed algorithm addresses particle dynamics as a function of time by inspecting the behavior of bunches of particles across the simulation. By combining this bunch lifetime analysis with a clustering algorithm estimation of which accelerated particles form the highest-energy beam becomes possible.

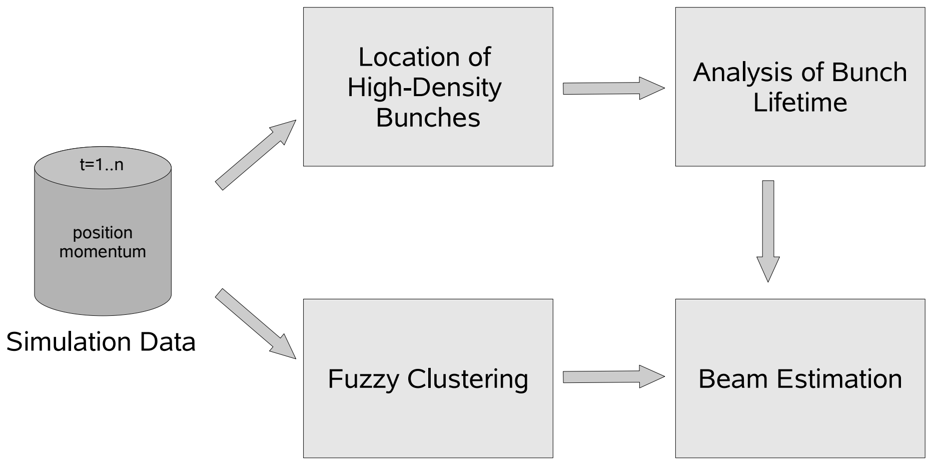


Figure 5.18: Processing pipeline for automatic beam detection.

Figure 5.18 illustrates the processing pipeline for automatic beam detection. The different parts of the pipeline are explained in the following subsections. First, the algorithm computes the location of high-density bunches of high-energy particles for each timestep via stationary point detection on probability density functions (PDF) of the particles along the laser propagation axis (x) (see Section 5.5.2). The so detected points are then organized in a graph to characterize the lifetime of potential high-energy bunches (see Section 5.5.3). Separate from the bunch lifetime analysis, a fuzzy clustering is executed for each timestep to detect particles forming a potential beam (see Section 5.5.4). Based on the information from the bunch lifetime analysis and the fuzzy clustering, the method estimates the most energetic beam (see Section 5.5.5). In the following sections a 2D particle dataset is used to illustrate the different steps of the method (see Table 5.1).

5.5.2 Locating High-density Bunches

The beams of interest are characterized by high density of high-energy particles in small spatial regions. The objective of the first analysis step is to identify potential groups of particles that exhibit this desirable property. The algorithm, therefore, first computes for each recorded timestep of the simulation a histogram of the particle distribution in the x-direction with density-peaks indicating locations of potential beams.

Particle energy is roughly proportional to the momentum in the x-direction (px). The algorithm, therefore, does not use the entire particle dataset, but instead considers only those particles with $px > 1e10$. This threshold eliminates low-energy particles — greatly reducing the workload for the analysis — without compromising the beam detection procedure.

Having identified the relevant particles, the algorithm estimates the PDF in x of the particles, $f(x)$, describing the density distribution of accelerated particles within the simulation window. The algorithm then computes maximal extrema points (maximal turning point or relative maximum) $x_maxima = \{X_0,...,X_k\}$, by computing $df(x)/dx$, for $f(x)$ changing from positive to negative [147]. Differentiability is guaranteed by using a Gaussian smoothing kernel. The kernel bandwidth of the filter is chosen using the following rule of thumb: 0.9 times the minimum of the standard deviation and the interquartile range divided by 1.34 times the sample size to the negative one-fifth power, as suggested by [148](pp 48-49). The maximal extrema points represent potential beam-point candidates for the next step of the analysis.

Figure 5.19 shows the x-location of beam-point candidates for a single timestep of the example dataset A (see Table 5.1). Note that the beam-point candidates correspond to high-density regions indicated by peaks of the smoothed PDF.

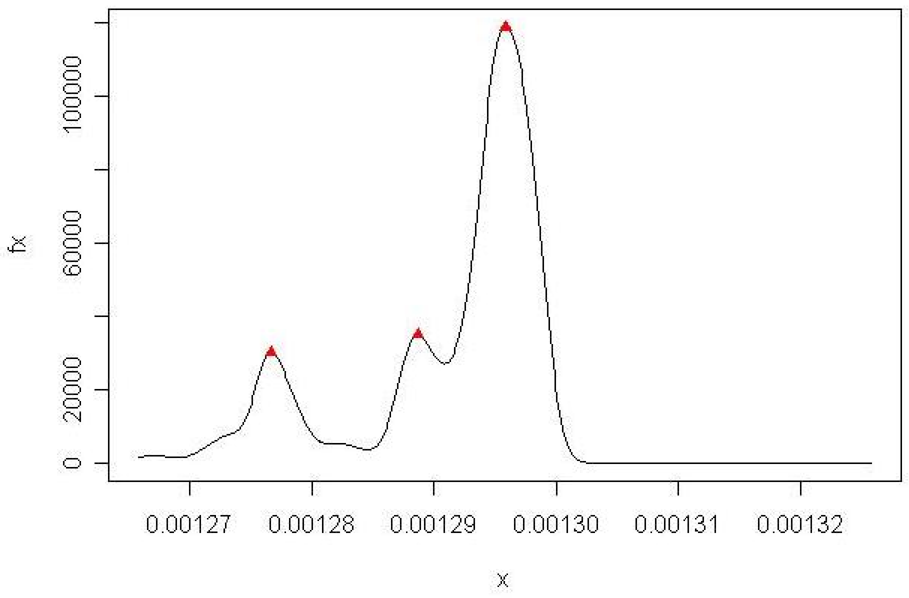


Figure 5.19: Locating high-density bunches for dataset A at a single timestep. The detected beam-point candidates (red triangle) define peaks of the smoothed PDF.

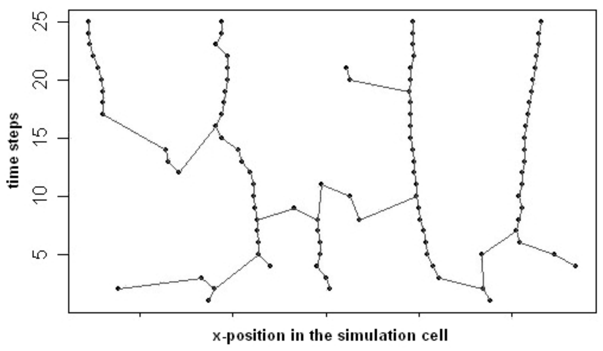
5.5.3 Analysis of Bunch Lifetime

After identification, the maximum extrema points of the different timesteps — each representing a potential beam-point candidate — are organized in a graph. Each node n of the graph represents a beam-point candidate. The edges of the graph then define corresponding beam-point candidates indicating the lifetime of high-density particle bunches.

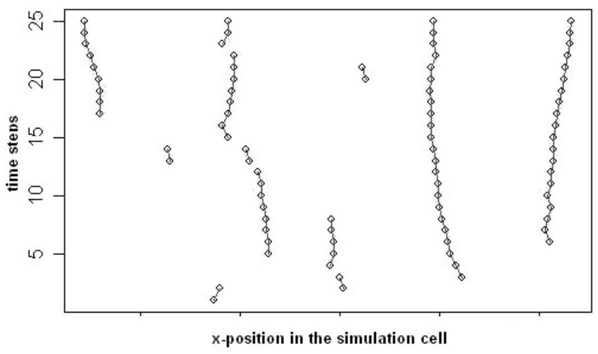
To construct this so called *bunch lifetime diagram*, the algorithm first constructs an incidence matrix for a graph by discretizing x axis into l bins. Bins that contain a beam-point candidate are assigned the value 1 whereas all other bins are set to 0. This process is repeated for each timestep t defining a set of feature vectors $fv(t) = b(t)_1, b(t)_2, ..., b(t)_l$, with $b \in \{0,1\}$, describing at which relative x locations a high-density bunch was found at timestep t. The feature vectors $fv(t)$ are stacked to define the temporal feature matrix:

$$fm = \begin{bmatrix} v(t_{min}) \\ v(t_{min}+1) \\ .... \\ v(t_{max}-1) \\ v(t_{max}) \end{bmatrix} = \begin{bmatrix} b(t_{min})_1 & b(t_{min})_2 & ... & b(t_{min})_l \\ b(t_{min}+1)_1 & b(t_{min}+1)_2 & ... & b(t_{min}+1)_l \\ ... & ... & ... & ... \\ b(t_{max}-1)_1 & b(t_{max}-1)_2 & ... & b(t_{max}-1)_l \\ b(t_{max})_1 & b(t_{max})_2 & ... & b(t_{max})_l \end{bmatrix}$$

Note, by merging the feature vectors $fv(t)$ to define the feature matrix fm, the different beam-point candidates from the different timesteps are aligned via their relative x location in the moving



(a)



(b)

Figure 5.20: Bunch lifetime diagram for dataset A. Particle history as a minimum spanning tree (MST) before (a) and after pruning (b). with likely branches and connected nodes.

simulation window. This is possible because the high energy particles travel at approximately the speed of light, i.e., approximately the same speed as the simulation window. The relative location in x of a beam-point candidate is, therefore, expected to be relatively stable.

To identify corresponding beam candidates — i.e, define the edges in the graph compromising the bunch lifetime diagram —, the algorithm then executes a minimum spanning tree (MST) algorithm [149], that penalizes connections among nodes in the same timestep (see Figure 5.20 a).

As described above, beam-point candidates are expected to be relatively stable in x_rel, i.e., their relative location in x within the simulation window. The approximate size of a beam, as suggested by the physicists, is typically no larger than $d = 2\mu$m for these parameters. Edges longer than $d = 2\mu$m, hence, define false connections between different bunches. In the next step, the created graph is, therefore, pruned by eliminating edges that are too long (i.e., longer than $d = 2\mu$m). After this pruning, the graph is guaranteed to contain only short edges and only edges that connect nodes (i.e., beam-point candidates) between different timesteps. Assuming that the time lag between recorded timesteps (i.e., dumps) is small enough to capture physical phenomena such as, bunching and dephasing, — a property selected by the scientists executing the simulation — the algorithm then also eliminates disconnected nodes. Figure 5.20 b, shows the resulting bunch lifetime diagram for dataset A, conveying the temporal history of only the most likely candidates (compare Figure 5.20a and b, before and after pruning).

The bunch lifetime diagram as defined here describes the temporal history of temporally stable, high-density particle regions defining potential particle beams. The bunch lifetime diagram is guaranteed to contain: i) only edges connecting different timesteps, ii) only short edges that are no longer than the expected beam size, and iii) only nodes that are connected to at least one other node. As illustrated in Figure 5.20b, a bunch lifetime diagram usually contains several independent graphs — i.e., graphs not connected via an edge to any other graph — each of which describes the lifetime of a different potential particle beam.

5.5.4 Fuzzy Clustering

As discussed in Section 5.5.2, a high particle density is not sufficient to identify a high quality beam, but both the spatial and momentum features play an important role in classifying a bunch of particles as a beam of interest.

Unsupervised algorithms are appropriate for data mining applications, where the information content of a large database is not known beforehand, but can emerge during the partitioning process. Without supervision, non-hierarchical clustering methods can use an optimization model to classify inter-point distances and dissimilarity data. The objective is to minimize total dissimilarity amongst all objects and the corresponding most representative objects.

In this context, the here described approach focuses on searching for the primary beam particles, i.e., the beam of highest energy. This beam should be confined to a small spatial region, be compact in momentum space, and having high energy indicated by high px values. Classical clustering algorithms would try to assign each data point to exactly one cluster [150]. The problem of detecting particle bunches, however, requires relaxing this condition so that each particle has some graded or fuzzy membership in each cluster.

In this work, the R package `cluster` is used to perform data clustering. More specifically, the algorithm uses the `fanny` [151] function to identify two groups of particles in $(x, y, px, py$ space via fuzzy cluster analysis. To ensure equal weighting of the different data dimensions the spatial and momentum variables are first normalized before calculating inter-particle distances. The most frequent normalization strategy consists in the transformation of the original data such that the new feature set is guaranteed to have zero mean and unit standard deviation [152]. Dissimilarities between particles are the computed using squared Euclidean distances. Afterwards, fanny performs a fuzzy C-means clustering based on the distance matrix by minimizing the objective function

$$F = \sum_{v=1}^{k} \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} u_{iv}^{m} u_{jv}^{m} d(i,j)}{2 \sum_{j=1}^{n} u_{jv}^{m}}, \tag{5.1}$$

where v is a cluster, n is the number of observations, k is the number of clusters, m is the membership exponent, $d(i, j)$ is the dissimilarity between observations i and j, and u_{iv} is the membership

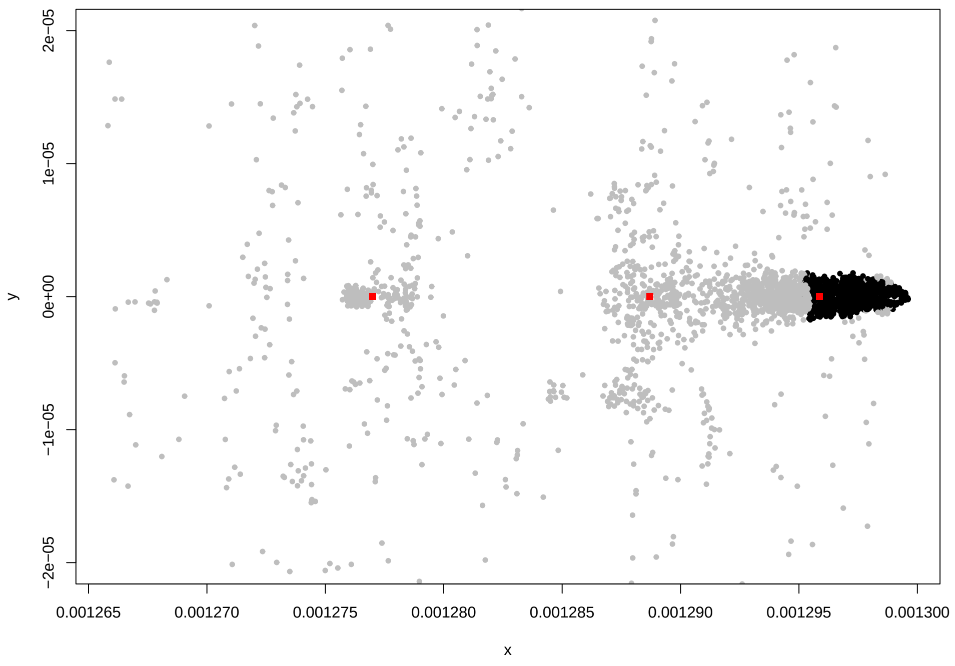


Figure 5.21: Overview of the results from the beam estimation for dataset A shown in physical space (x,y). The beam-point (squares) (see also Figure 5.19) are combined with the result from the fuzzy clustering. The beam particles are shown in black and the non-beam particles are shown in gray. Note, here only the particles with $px > 10^{10}$ used in the analysis are shown.

of observation i to cluster v [151]. The dissimilarity measure appears as an $L1$ norm; it finds medoids[11] instead of ordinary centroids. The minimization algorithm is based on direct application of the Lagrange multiplier approach with Kuhn-Tucker conditions [153].

5.5.5 Beam Estimation

To estimate the highest-energy bunch, the algorithm then compares the estimated-beam cluster, calculated for each timestep independently using fuzzy clustering, with the beam-points for the respective timestep included in the bunch lifetime diagram (i.e., those points at a timestep that remained after the pruning; see Section 5.5.3). If the most accelerated group of particles — which is expected to be at the rightmost cluster — and a beam-point overlap, then the algorithm concludes that the beam was found and returns the particles of the according cluster.

[11] *Medoid = Median-based centroid.* A medoid is that object of a cluster with the minimal average dissimilarity to all objects in the cluster. Similar to the mean, medoids are used in data clustering as representative objects for clusters, in particular in the context of data where a meaningful mean or centroid cannot be defined.

The bunch lifetime diagram combined with the clustering output yields two important pieces of information: (i) detection of the timestep containing high-energy particles by checking for overlapping; and (ii) estimation of a beam containing particles that behave similarly, according to their spatial coordinates and energy attributes.

Figures 5.21 and 5.22 a, provide an overview of the result of the beam estimation for the example dataset A. In both figures the cluster membership is narrowed to a value of 0.7, so that the cluster closer to the rightmost high-density point will contain only particles with a 70% probability to belong to this cluster. This idea is analogous to the search for the core particles in the multi-variate distribution given by (x, y, px, py) for the primary beam. The beam cluster is labeled in black in both Figure 5.21 and Figure 5.22 a. Physicists have noticed that the estimated beam completely enclose the high energy bunch of particles.

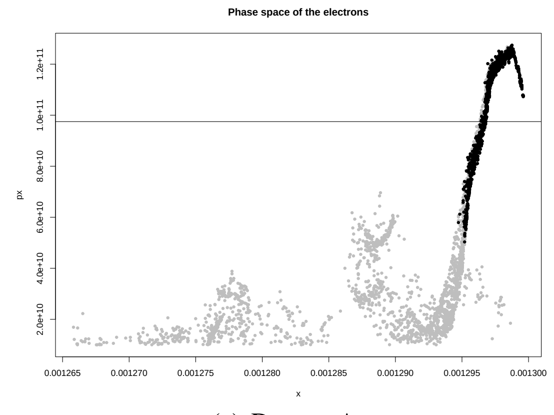
5.5.6 Validation

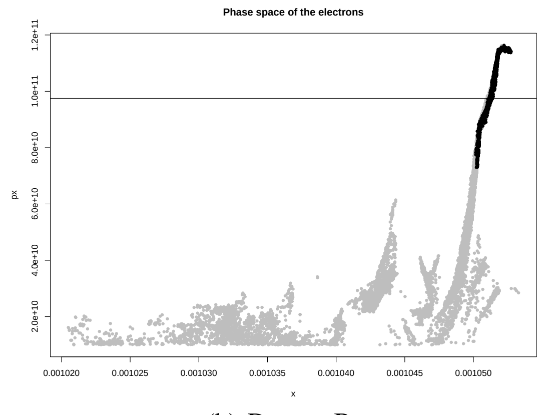
In this section the proposed method based on the bunch lifetime diagram and fuzzy clustering is evaluated for various 2D particle datasets. The proposed framework contains data processing and machine learning algorithms from R project [154, 128]. The R project, or simply R, is a free multi-platform software environment for statistical computing, containing useful packages for data analysis, visualization and machine learning. All computations were performed on a Dell Optiplex 755 Intel Core Duo 3GHz. The processing took ≈1-4 minutes for each timestep using 2 Gb RAM memory. To validate the described method, it is applied to four 2D particle datasets described in Table 5.1. The datasets vary in size as well as overall behavior.

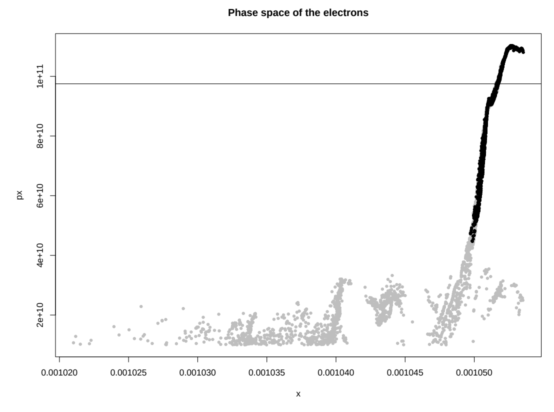
Dataset	Particles (10^3)	Timesteps	Total Size (Gb)
A	0.4	38	1.3
B	1.6	36	4.5
C	0.4	38	1.3
D	3.2	46	11

Table 5.1: Simulation datasets used for testing and validation of the proposed approach for automatic beam detection (particle data only).

Figure 5.22 shows the resulting phase space diagrams (at the time when the beam is most visible) using the proposed methodology for the four different 2D datasets. In practice a physicist is interested in finding single, distinct, and compact particle bunches of high momentum and low energy spread in the data.


(a) Dataset A


(b) Dataset B


(c) Dataset C

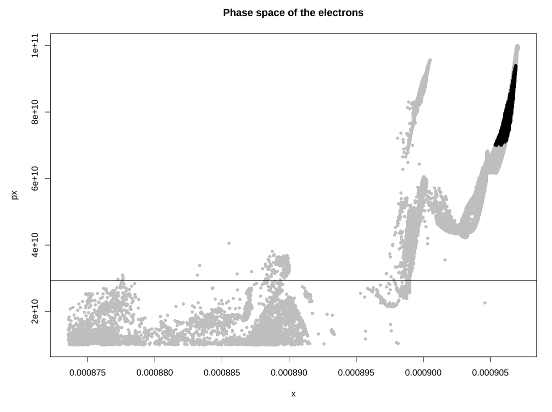

(d) Dataset D

Figure 5.22: Clustering results shown in a phase space $(x, px0)$ scatter plot for the four test particle datasets. All particles with $px > 1e10$ are shown in gray and the particles detected by the automatic beam detection are shown in black.

In a non-interactive manual analysis process, thresholding in *px* is common practice to define high-energy particles and the beam of interest. To define a beam, a researcher investigates movies of a variety of plots to determine an appropriate timestep and threshold values. As illustrated in Figures 5.22(a)- 5.22(c), for simulations A, B, C a researcher may choose, for example, a threshold of $px > 9.75e10$. In these examples only a single high energy beam is formed and a single threshold in *px* is sufficient to isolate the beam-particles. However, simulation D contains secondary structures, formed behind the beam and a single threshold in *px* is not sufficient to isolate the beam-particles (see Figure 5.22(d)). In this case, a single threshold that captures all beam particles would also enclose a large number of non-beam particles and, hence, result in a selection that is too large. To isolate the beam of interest multiple thresholds in *px*, *x* and possibly *y* would be required to isolate the beam of interest. One main deficiency of thresholding is that it is arbitrary and time-consuming, requiring manual inspection of the dataset.

Figure 5.22 demonstrates that the proposed method detects a single, distinct, bunch of high energy particles in all four datasets. Both thresholding and the proposed algorithm depict particles of the beam for dataset A, B, and C, but our method is capable of identifying subsets of condensed particles among the highly accelerated particles without requiring user interaction. For experiments where no high quality beam is formed, neither thresholding nor the proposed method can accurately detect beam particles as shown for dataset D. In dataset D, the most condensed structure is located on the depression between the first and second peak ($px = \approx 4.5e10$), as shown in Figure 5.22(d). While the automatic beam detection algorithm detects a single, spatially distinct particle bunch of high energy, it is driven towards finding distinct bunches of high energy and the clustering result encloses the second highest quality bunch.

The proposed approach presents promising results towards automated analysis of laser wakefield particle simulations. The method is capable of extracting particle beams and isolating timesteps without user interaction, in contrast to manual thresholding wherein an expert is required to manually investigate the whole data in order to identify the beam of interest. With respect to the described implementation, the main limitations of the introduced method are: i) the computational performance is limited due to the serial implementation in R, ii) due to the performance limitations the implementation has only been tested for 2D particle data, and iii) the method in the described form is driven towards high energies.

5.6 Automatic Beam Path Analysis in Laser Wakefield Simulations

As discussed previously in Section 5.5, auxiliary mechanisms are needed to automate the most time-consuming parts of the analysis in order to enable analysis of large databases of simulation data. The goal of the automatic beam path analysis algorithm described here is to enable: (i) automatic detection of multiple particle bunches in a single simulation, not just the one of highest energy, (ii) definition of beams as a temporal feature of the data, i.e., the beam-classification should be based on the complete temporal history of the particles that form them, (iii) analysis of the temporal evolution of particle beams, and (iv) the algorithm should be efficient, i.e., be able to complete the analysis within at most several minutes rather than hours. High performance is particularly critical to ensure that: (i) the algorithm can be used for the analysis of a large number of datasets, and (ii) to make it a practical tool for everyday research.

To achieve high performance the proposed algorithm employs an efficient analysis pipeline aimed at quickly reducing the amount of data that needs to be considered during the analysis. In the initial analysis steps the algorithm first analysis the different timesteps independently to gather additional information about the particle bunches and to reduce the workload for later analysis steps. Efficient grid-based methods are used to enable fast identification and segmentation of particle bunches at individual timesteps independent of the size of the underlying data. After completion of the initial timestep analysis, the algorithm then merges the results from the individual timesteps and computes for each detected bunch a reference path describing its temporal evolution. To enable accurate classification of particle bunches the algorithm then completes the analysis by computing a set of distance fields describing the distance of a particle's path to a bunch. Based on these path distance fields a user can then effectively define which particles from a bunch of interest. To achieve good performance the algorithm uses HDF5-FastQuery [122] — the same data management library used to enable high-performance visual data exploration (see Section 5.4) — for fast: (i) computation of 3D conditional histograms used for the segmentation of particle bunches, (ii) evaluation of range queries, and (iii) evaluation of ID-queries used for particle tracing (see also Section 5.4.3.2).

The following parts of this section are structured as follows. Section 5.6.1 discusses how a particle beam is defined and describes additional assumptions the algorithm is making in the beam detection process. By defining the feature of interest Section 5.6.1 also describes the basic requirements the analysis algorithm has to fulfill. Section 5.6.2 then gives an overview of the complete analysis pipeline. The different steps of the analysis pipeline are described in Sections 5.6.3- 5.6.6. In Section 5.6.7 the algorithm is applied to a set of different 2D and 3D particle datasets demonstrating the validity and effectiveness of the algorithm in the context of real-world examples. The integration of the analysis and the visualization is then discussed later in Section 5.7. Afterwards, Section 5.8.2 describes a more elaborate example illustrating the use of the automatic beam path analysis in combination with VisIt for: (i) comparison of different particle bunches, (ii) evaluation of bunch quality, as well as (iii) analysis of the temporal evolution of particle bunches.

The specific new contributions of the work described in this section are:

(i) A novel pipeline for automatic detection and analysis of particle bunches in laser wakefield particle accelerator simulations is described. As part of the pipeline the following new methods are introduced:

    a) An efficient method for detection of particle bunches at single timesteps of a simulation based on region growing (see Section 5.6.4).

b) A method for combining particle bunch classifications from different timesteps to define a single consolidated description of multiple particle bunches detected in a single simulation (see Section 5.6.5).

c) A novel method for classification of particle bunches based on the temporal paths of the particles that form them. In this process the analysis also derives additional information about the different temporal phases of particle bunches, e.g., when was a bunch formed or accelerated (see Section 5.6.6).

(ii) To achieve good performance the algorithm makes use of FastBit for efficient data access and particle tracing. In this context, this thesis introduces new functions for computation of three-dimensional (3D) conditional histograms that were specifically developed for this work and integrated into FastBit.

(iii) The proposed method is applied to a variety of 2D as well as 3D particle data sets to demonstrate its validity and effectiveness (see Section 5.6.7). A more elaborate example illustrating the use of the proposed methods for comparison of particle beams is presented later in Section 5.8.2.

(iv) Section 5.6.8, then examines the performance of the proposed analysis pipeline and its different components.

5.6.1 Feature Definition and Assumptions

The beam path analysis is aimed at detecting particle beams that are characterized as follows:

- **F1:** A beam is defined by a compact bunch of accelerated particles (i.e., particle with high *px* values) condensed in x, y px, py space. In 3D simulations also z and pz. This means: i) the particle bunch is coherent in physical as well as momentum space, and ii) the particles forming the bunch have high *px* values (see, e.g., Figure 5.25b).

- **F2:** In the simulations the laser pulse is centered in the plasma at $y = 0$ and $z = 0$ and moves along the x axis. The laser pulse traveling through the plasma creates waves in the plasma which in turn accelerate the particle beams (see Figure 3.8).

- **F3:** Particles are accelerated in the same direction as the laser pulse, here x direction.

- **F4:** While being trapped in the wave the particles forming a bunch are accelerated over a period of time until they eventually outrun the wave and decelerate again. These particle bunches are therefore present over a period of time so that the bunches themselves define a temporal coherent feature of the data.

- **F5:** The particle waves created by the laser pulse will manifest as peaks in x, y, px space (see, e.g., Figure 5.25b).
- **F6:** In the analysis py and pz are used to assess the quality of particle beams as well as to define the distance of a particle to the beam. The algorithm does, however, not use py and pz to detect condensed particle bunches at single timesteps, because a particle beam is usually not defined by a single condensed group of particles in y/py (or z/pz) space but due to the characteristic oscillating transverse motion of accelerated particles constitutes as two (or more) condensed groups of particles in this space (see, e.g., Figure 5.50 b and c). The transverse momenta py and pz are hence symmetric variables centered around zero.
- **F7:** Particles may become trapped in different periods of the plasma wave. In practice, several particle beams may, therefore, form in different periods of the wave and possibly coexist at the same time. Furthermore, after a beam has decelerated (see F4) a new beam may form later in time within the same period of the wave, i.e., at different times of the simulation one may find different particle beams at similar locations within the simulation window.

Further, the algorithm makes the following assumption:

- **A1:** Within each main peak in $x/y/px$ space exists only one high quality bunch of interest at a time. As will be shown later, in cases where this assumption may not be true the algorithm will make an automatic, implicit decision and retrieve the bunch that defines the highest density feature within that peak for the longest period of time. This bunch is often also the one exhibiting the highest acceleration.

As described in Section 3.2.3, only a fraction of all timesteps of a simulation are saved to file. Even though the temporal resolution of the saved data is usually not high enough to resolve the oscillation frequency of the wave, it is ensured by the user to be sufficient to resolve acceleration and dephasing (i.e., the process during which a beam loses its coherency). It is, therefore, save to assume that:

- **A2:** Between two consecutive timesteps a particle bunch does not disappear while a new bunch appears at the same location (see also F7).

5.6.2 Overview of the Algorithm

Particle beams are small compared to the total size of the data. The beam path analysis pipeline is designed to enable quick data reduction and efficient detection of particle bunches. Using state-of-the-art data management based on HDF5 [120], H5Part [123, 124] and FastBit [51, 122] the algorithm is able to efficiently extract the portions of the data relevant for the analysis. In order to be able to accurately detect and classify particle beams the analysis has to consider information of the complete timeseries. The algorithm initially analyze each timestep separately to collect information on the particle beams. This information allows for significant reduction of the amount of data that need to be consider in the later analysis of the temporal particle paths.

Figure 5.23 provides a high-level overview of the general structure of the analysis pipeline. The main steps of the analysis pipeline are explained in detail in Section 5.6.3- 5.6.6. The analysis distinguishes between two different types of particles. *Candidate particles* are all particles that were ever detected by the analysis as being part of a particular bunch, i.e., particels that met the bunch criteria described in Section 5.6.1 at least once in any timestep. *Reference particles* are a subset of candidate particles found with a high frequency, i.e., particles with a high degree of temporal persistence within a bunch.

For illustration purposes a medium-sized 2D dataset is used to describe the different steps of the analysis. The later validation of the analysis then also presents results for a large 3D dataset (see Section 5.6.7). The example dataset shows a fairly complex acceleration behavior

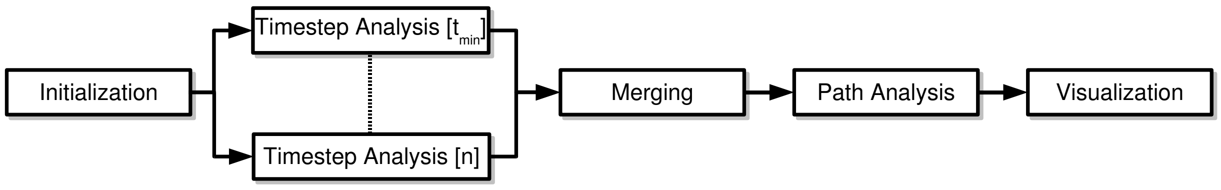


Figure 5.23: **Overview of the algorithm design:** The initialization step identifies which timesteps need to be analyzed. Afterwards, each timestep is analyzed independently in order to identify the most prominent particle bunch at each timestep. In the merging step the algorithm then combines the information from the different timesteps and: (i) identify the number of detected bunches; (ii) compute for each bunch a set of candidate particles; and (iii) compute for each bunch a discrete frequency function describing how often the individual candidate particles were found to be part of the bunch. In the particle path analysis all candidate particles are traced over the complete timeseries and for each bunch a reference path is computed. Based on the reference path the algorithm then identifies the different phases of a bunch, such as acceleration and deceleration, and define for each candidate particle the distance to the bunch. After completion of the analysis process the user investigates the analysis results using dedicated visualization methods.

and contains bunches of different quality at high as well as low energy levels (see Table 5.2 dataset C for more details). The analysis detects in the example dataset two main particle bunches referred to as *first* and *second* in the following. The first bunch is also detected first by the analysis indicating that this bunch is formed earlier in time than the second bunch.

5.6.3 Initialization

The main purpose of the initialization step is to calculate the minimum timestep t_{min} indicating which timesteps are relevant for the initial timestep analysis. As described earlier, a particle beam consists only of accelerated particles, i.e., particles with high px values. At early timesteps of the simulation no particles at a sufficient level of acceleration exist since the plasma waves —induced by a laser pulse traveling through the plasma — are just forming. These timesteps are hence not relevant in the initial timestep analysis described later in Section 5.6.4.

To prevent unnecessary calculations and to avoid detection of small bunches at timesteps containing very few accelerated particles the algorithm employs a minimum timestep t_{min}. Often the user has already a good understanding of when the first particle beams are forming in the data based on earlier analysis. The algorithm, therefore, allows the user to either: (i) define t_{min} manually as input parameter of the analysis or (ii) have the algorithm estimate a good value for t_{min} automatically.

In cases where the user has no prior knowledge of the value for t_{min} the algorithm determines t_{min} as follows. First, the analysis computes for each timestep t the number of particles $h(t)$ that satisfy the condition $px > 10^{10}ms^{-1}$. The same (user-defined) condition is also applied in the later timestep analysis and ensures that only particles with a sufficient acceleration level are considered. Section 5.6.4.1 provides a more detailed explanation of the employed condition. Based on $h(t)$ the method then computes the average number a_p of accelerated particles at timesteps t with $h(t) > 0$, i.e., $a_p = \frac{\sum_{t=0}^{n} h(t)}{m}$, with n being the total number of timesteps and m being the number of timesteps with $h(t) > 0$. t_{min} is then defined as the first timestep t with $h(t) > a_p$. This ensures that the analysis considers only timesteps ($t \geq t_a$) with a sufficient amount of accelerated particles at which one can expect to find a well defined bunch that meets the criteria defined in Section 5.6.1.

This automatic approximation works especially well in those cases where the number of accelerated particles $h(t)$ reaches a relatively stable state. In complex cases where the number of accelerated particles $h(t)$ varies largely over time the described approach may, however, lead to a suggestion of a too late timestep. After the initialization is complete the beam path analysis performs for each relevant timestep the analysis described in the next section.

5.6.4 Timestep Analysis

The initial timestep analysis is aimed at identifying the most prominent particle bunch at each timestep, i.e., the bunch with the highest px value. The goal is to identify a group of candidate particles for the different bunches, i.e., a set of particles that potentially belong to a bunch. Focusing only on these much smaller sets of candidate particles greatly reduces the workload for the later particle tracing. The timestep analysis is performed independently for each timestep. and consists of two main steps: i) data preparation and ii) bunch segmentation described in Sections 5.6.4.1 and 5.6.4.2 (see Figure 5.24).

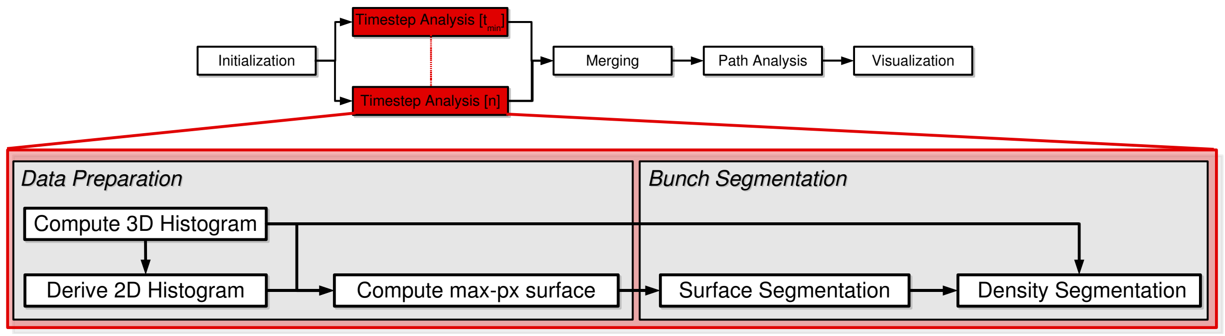


Figure 5.24: **Overview of the timestep analysis:** The analysis uses a grid-based segmentation approach to identify particle bunches of interest at each timestep. First, the analysis computes a 3D histogram in $x/y/px$ of all accelerated particles ($px > 10^{10}ms^{-1}$). From the 3D histogram the 2D histogram in x/y is derived. Based on the information of the histograms the function p_{max} is computed defining the maximum px value found at each (x,y) location of the used analysis-grid. Segmentation of the surface defined by p_{max} allows for identification of the region of interest (ROI) in physical space (x/y). A second density-based segmentation approach then identifies the most condensed particle bunch within the ROI.

5.6.4.1 Data Preparation

The data preparation step is executed once for each relevant timestep (i.e., $t \geq t_{min}$) and is aimed at initializing all data structures needed for the later bunch segmentation step (see Section 5.6.4.2). The bunch segmentation described later in Section 5.6.4.2 uses two grid-based data structures in the bunch segmentation step of the analysis pipeline: i) a 2D analysis-grid defined in the physical domain (x/y); and ii) a 3D analysis-grid defined in $x/y/px$ space. The 3D analysis-grid describes a scalar field defining how many particles belong to each grid point, i.e., the analysis computes a 3D histogram defined over the domain $x/y/px$. Associated with the 2D analysis-grid is a second scalar field describing the maximum px value of all particles associated with each point of the grid. In case of a 3D dataset the user may use the radial distance $d_r = \sqrt[2]{y^2+z^2}$, y or z instead

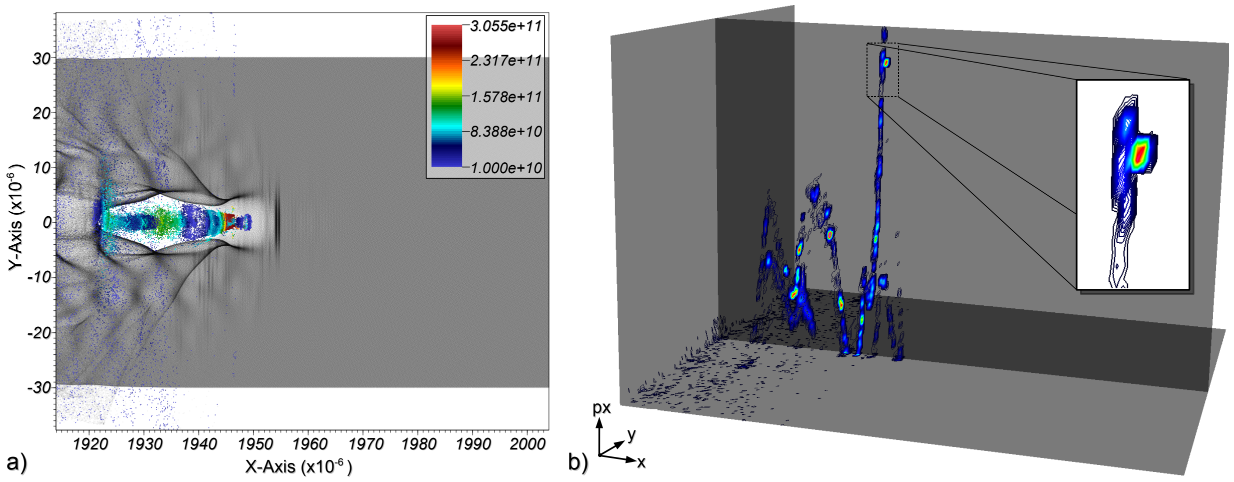


Figure 5.25: Visualizations of timestep $t = 40$ of the example 2D dataset consisting of $\approx 2.4 * 10^6$ particles. a) All particles shown in physical space (gray) and all particles that satisfy the condition $(px > 10^{10})ms^{-1}$ ($\approx 1.7\%$ of all particles) colored according to px. b) Iso-contours of the particle density shown on three slicing planes to illustrate the basic structure of the 3D particle density in x, y, px. The iso-contours are colored according to their value with black/blue being low, green/yellow being medium, and red being high density. The inset plot shows a close-up view of the main region of interest containing a condensed particle bunch.

of y as second dimension of the 2D and 3D analysis grid. Since the example dataset used to illustrate the different steps of the algorithm is a 2D dataset the following parts of this thesis refer for consistency to the radial dimension always as y.

3D Histogram Computation: In the 3D histogram computation the algorithm computes: (i) the count of each bin of the 3D histogram, and (ii) a set of bit vectors that indicate which particles are associated with each bin of the 3D histogram.

According to **F1** (see Section 5.6.1) a particle bunch of interest will consist only of accelerated particles. The expected wake oscillation is up to $px = 10^9 ms^{-1}$ and the particle beams of interest should be observed near $px = 10^{11} ms^{-1}$. At each timestep the analysis therefore considers only particles that satisfy the condition $px > 10^{10} ms^{-1}$ which ensures that only particles with a momentum in x direction above the base acceleration of the wave are used while including all particles that are potentially of interest. As illustrated in Figure 5.25a, this condition significantly reduces the amount of raw data that needs to be accessed by the algorithm — i.e, usually to only $\approx 1-3\%$ of all particles —, significantly improving the performance of the analysis.

To be able to identify condensed particle bunches at single timesteps the algorithm uses a 3D histogram of x, y, and px with a resolution of usually 100 bins per variable and the described condition of $px > 10^{10}ms^{-1}$. Consistent with **F6** (see Section 5.6.1), the algorithm does not consider the transverse momenta py and pz at this stage of the analysis. In x and y the 3D histogram is computed over the entire extension of the simulation window at the current timestep while using the base threshold of $px = 10^{10}ms^{-1}$ as lower bound in px. The size of the simulation window is constant in x and y over time, i.e., one can correlate bins of 3D histograms from different timesteps directly via their index in (x, y). As illustrated in Figure 5.25b, the 3D histograms will always be sparse with accelerated particles appearing only behind the laser pulse — located roughly in the center of the simulation window — and most condensed around $y = 0$.

The later bunch segmentation (see Section 5.6.4.2) identifies those bins of the 3D histogram containing a particle bunch of interest. To be able to efficiently access the data of the selected bins this thesis introduces dedicated 3D histogram functions that compute the actual counts of the 3D histogram as well as a set of bit vectors indicating for each non-empty bin which particles belong to the according bin. These bit vectors are compressed and provide a memory and computationally efficient way for creating and storing the inverse mapping from a 3D histogram to the original data. Bit vectors can efficiently be merged using bitwise OR operations, i.e., bit vectors enable also efficient access of the data of many bins at once. Since the 3D histograms are always sparse the memory overhead due to the bit vectors is in practice small, usually $\approx 2-5MB$ per timestep depending on the data distribution and size of the dataset.

Maximum *px* Function Computation: To be able to identify regions of high particle acceleration the algorithm computes, based on the information of the 3D histogram, the maximum px function:

$$p_{max}(x, y) = max(px(x, y)), \tag{5.2}$$

defined over the physical domain (x/y). This function associates with each point in physical space the maximum px value of all particles found at that location. The algorithm approximates this function by defining a 2D analysis-grid in physical space with the same resolution as the 3D histogram, i.e, usually 100 bins per variable. For each (x, y) column of the 3D histogram the algorithm then computes the maximum px value by identifying the first bin (from top to bottom) with a count larger than zero using the upper bin-boundary in px as reference. In cases where outlier behaviors imposes a problem one could, e.g., also use the px value of the highest-density bin of each (x, y) column as reference instead.

A density-based filtering of p_{max} ensures that the analysis is always focused on the areas of highest particle density. First, the 2D histogram in x/y is derived from the 3D histogram by adding up the counts of the each (x, y) column of the 3D histogram. For the 2D histogram only the counts and no bit vectors are computed. The p_{max} function values of all points of the 2D

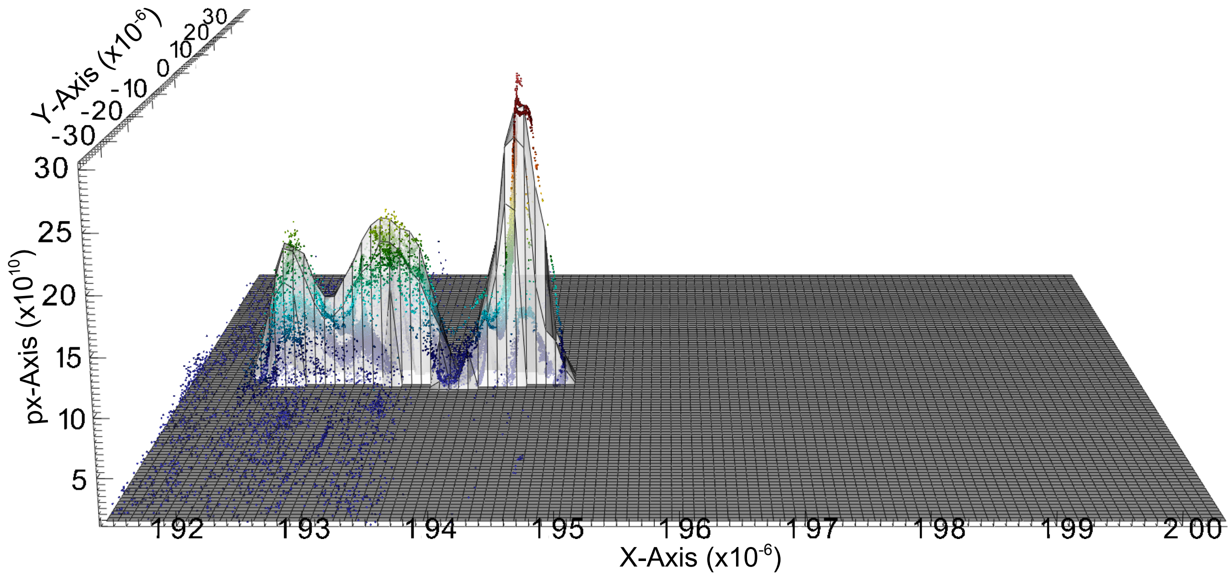


Figure 5.26: Surface plot showing the approximated maximum px function p_{max} defined over the x/y domain at timestep $t = 40$. Each point of the surface represents one column of the 3D histogram in (x, y) and is located in its center. The z coordinate of each point is defined based on the p_{max} function value computed for the according grid-point. In addition to the p_{max} surface all particles with $px > 10^{10} ms^{-1}$ are shown colored according to px.

analysis-grid with an associated particle density of less than 3% of the maximum 2D density are set to the minimum function value. This density-based filter removes small peaks in the outer sparsely populated parts of the simulation window while preserving the main peaks of interest in the center.

The algorithm approximates the value of the maximum px function p_{max} by smoothing the computed p_{max} function values using the following smoothing-kernel:

$$\begin{bmatrix} 0.075 & 0.075 & 0.075 \\ 0.075 & 0.4 & 0.075 \\ 0.075 & 0.075 & 0.075 \end{bmatrix}$$

derived through empirical study. Figure 5.26 illustrates the structure of the derived surface at timestep $t = 40$ of the example dataset. One can see that the derived surface approximates the general structure of the function well. The maximum level of the surface is expected to be lower than the peak px of the particles due to the used smoothing kernel. Peaks with a low support are removed by the 2D density filter.

5.6.4.2 Bunch Segmentation

The bunch segmentation is aimed at identifying a single particle bunch of interest at a given timestep. This process is initially executed only once per timestep. Additional bunch segmentations may then be performed at selected timesteps during the later merging process. The segmentation of a particle bunch is performed in a two-step process. The algorithm first performs a 2D segmentation based on p_{max} to identify the region of interest (ROI) in physical space (x/y). A second 3D density-based segmentation in $x/y/px$ space then identifies the most condensed particle bunch within the ROI.

The segmentation itself is based on region growing. While the neighborhood and stop criteria are different for the 2D and 3D region growing the basic algorithm is similar. In the region growing the analysis maintains two lists: i) a list of selected points, and ii) a list of candidate points. The first list defines all points that have been identified as being part of the ROI and the second list contains all points that potentially belong to the ROI but still need to be checked. Initially the list of candidates contains only a single seed-point and the list of selected points is empty. The segmentation function iterates through the list of candidates until no more candidates remain. For each candidate the segmentation checks whether it satisfies a set of criteria (the so-called stop criteria). If the candidate point meets the criteria then it is moved to the list of selected points and its neighbors are added to the list of candidates, otherwise it is removed from the list of candidates.

To identify the ROI in physical space the algorithm first executes a 2D segmentation based on p_{max}. Using the global maximum of p_{max} as seed-point a 2D region growing is used to identify the region in physical space associated with the seed. In this process the region growing uses the 4-neighbor stencil (see Figure 2.11) to define the neighbors of a given point of the 2D analysis-grid. A given candidate point is added to the list of selected points if it does not define a minima in x-direction of p_{max} neither in $x/y/px$ nor x/px space. The second condition prevents bleeding of the segmentation into secondary peaks of p_{max}. In this particular case the segmentation does not use the strict definition of a minimum but define a point to be a minimum if it does not have any neighbor in x-direction with a p_{max} value smaller than its own. The segmentation process is designed to ensure that the ROI in physical space is closed and contains no holes.

The selection defined by the 2D segmentation is usually too large, i.e., it includes many particles in the vicinity of the bunch that do not actually belong to it. To ensure that the segmentation result is always centered around the most condensed bunch a secondary 3D density-based segmentation is performed within the given 2D ROI. In this process the segmentation uses the bin of the 3D histogram that defines the highest density within the identified region in physical space

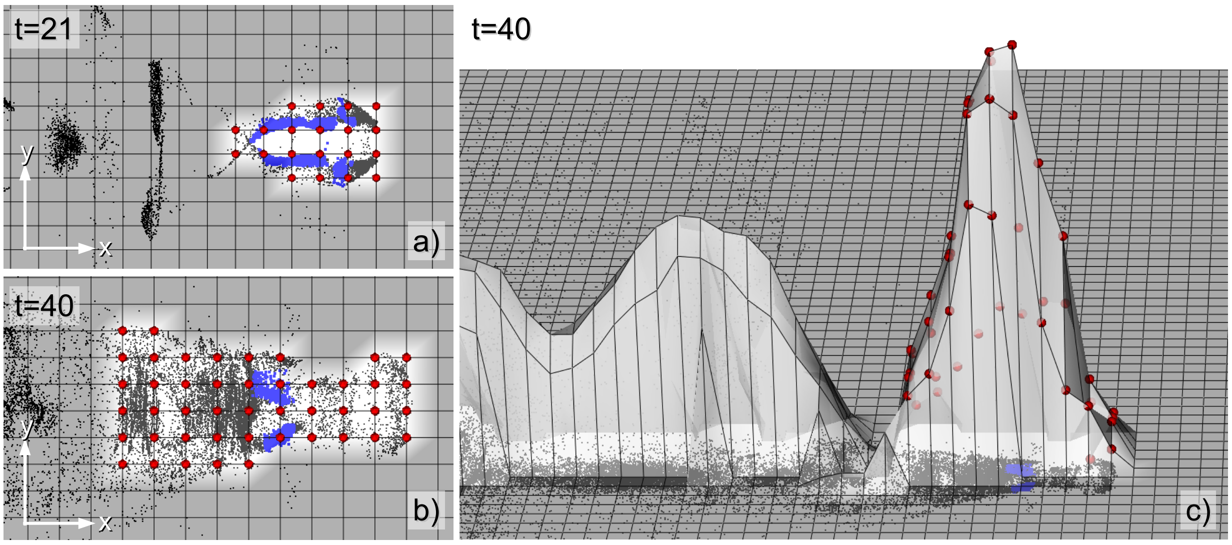


Figure 5.27: a) Overview of the segmentation process as performed at timestep $t = 21$ of the example dataset. The grid of the segmentation surface is shown in black. The points of the surface –each corresponding to one x/y column of the 3D histogram– selected by the initial surface segmentation step are shown in red and the corresponding area is highlighted in white. All particles with $px > 10^{10}ms^{-1}$ are shown in black and all particles selected by the final 3D density-based segmentation are shown in blue. b) Same as *a* but for timestep $t = 40$. c) Same as *b* shown as a 3D rendering. The surface is the same as shown in Figure 5.26. The initial surface segmentation defines the region of interest (ROI) in physical space. The density-based segmentation then identifies the most condensed group of particles within the ROI.

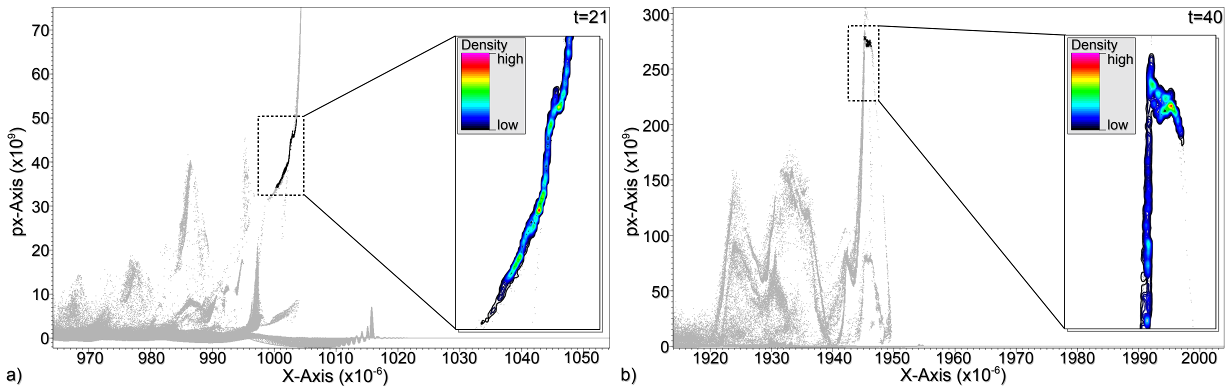


Figure 5.28: Example segmentation result at timestep $t = 21$ (a) and $t = 40$ (b) shown in x/px space (see also Figure 5.27). The particles identified by the segmentation process are shown in black, all other particles are shown in gray. The inset plots show a close-up view of the detected bunches with additional iso-contours of the particle density colored using the indicated color mapping. Due to the density-based segmentation approach the algorithm is able to detect condensed particle bunches at high (b) as well as low (a) energy levels.

as seed. The 3D region growing employs the 26-neighbor stencil (see Figure 2.11) and stop if the density value of a candidate point is below 20% of the density-value of the seed. In the later merging step of the analysis pipeline the IDs of the selected particles are used to identify corresponding particles at different timesteps. After completion of the segmentation the algorithm therefore loads the IDs of the particles associated with the selected bins. The bit vectors computed together with the 3D histogram enable the algorithm to directly access the according particle IDs.

Figure 5.27 illustrates the segmentation process for two selected timesteps of the example 2D dataset. Figure 5.28 then provides an overview of the result of the bunch segmentation at the according timesteps. As will be shown later, the bunches shown here in fact define two different main bunches of interest in the example dataset.

The bunch segmentation algorithm has several significant characteristics relevant for the later merging and path analysis. First, the bunch segmentation is always centered around the highest density and hence centered around the main bunch of interest. This ensures that the particles that define the "core" of a bunch of interest are always found at a higher frequency, i.e., they are found as being part of the bunch at more timesteps than particles that are in its larger vicinity. Second, like most segmentation algorithms, the bunch segmentation described here may suffer from under-segmentation (too large a selection) and over-segmentation (too small a selection) errors. The potential effects of under- and over-segmentation are taken into account in the later particle path analysis so that the beam path analysis deals robustly with these errors. In the context of a cluster ensemble these effects are in some sense even desirable since they ensure diversity. In the context of the complete analysis pipeline under-segmentation errors simply lead to inclusion of particles distant to the actual bunch in the list of potential candidates. These particles are identified later in the particle path analysis and do not affect the quality of the analysis. The two-step segmentation process furthermore ensures that in all cases only particles within a relatively small region in physical space are selected so that the amount of potentially improperly selected particles is in general low. The potential effects of over-segmentation errors are accounted for when selecting a set of reference particles for each bunch described later in Section 5.6.6.2.

5.6.5 Merging

In the merging step of the analysis pipeline the information derived from the individual timesteps is combined to: (i) identify the number of detected bunches, and (ii) define a consolidated description of each detected bunch consisting of a set of candidate particles and a discrete frequency

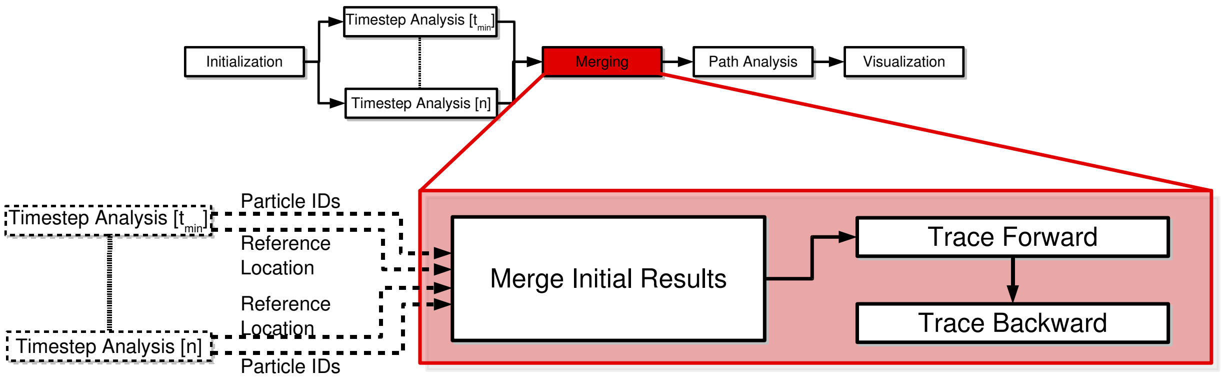


Figure 5.29: **Overview of the merging process:** The merging step of the algorithm first correlates the detected bunches via their given reference location in physical space. Instead of one bunch per timestep one now has a set of bunches each existing over a series of timesteps. In the initial timestep analysis the algorithm identified the single most prominent bunch at each timestep. In practice several bunches of interest may coexist at the same time, e.g., one bunch in the first and another one in the second period of the wave behind the laser pulse. The algorithm, therefore, traces the identified bunches forward and backward in time — performing additional segmentations at the individual timesteps — to complete the information about each bunch and ensure accuracy of the initial analysis. During the forward-tracing initially separated bunches may also be identified as actually being the same and merged. Each particle detected at least once as being part of a particular bunch is added to the according list of candidate particles. In this process the merging function counts for each candidate particle how often it was found as being part of a specific bunch.

function. The term frequency is used here to describe for each candidate particle how often — i.e., at how many timesteps — it was found as being part of the bunch (see Figure 5.29).

The problem one has to solve in the merging is twofold; the algorithm has to solve: (i) a correspondence problem to identify which segmentations from the different timesteps define the same bunch, and (ii) a combination problem in order to define a single consolidated description of each of the different bunches. From the initial timestep analysis the merging step receives two different inputs from each timestep: (i) a reference location describing the principle location of a bunch —here defined as the x-index $x_i(t)$ of the segmented maximum of the function p_{max}—, and (ii) a list of particle IDs describing which particles were identified as being part of the detected bunch.

To solve the correspondence problem the merging relates bunches from different timesteps via their given reference location $x_i(t)$. As explained earlier in Section 3.2.3, the simulation employs a moving window approach in which the simulation window is moved along the cavity while the laser pulse is traveling through the plasma. The simulation window, as well as the accelerated particles, move roughly at the same speed as the laser pulse, i.e., speed of light c. Furthermore, the size of the simulation window in x is constant. The relative $x-$location of maxima in p_{max} is therefore expected to be relatively stable, i.e., their maximum movement in x-direction between two consecutive timesteps is restricted by the difference of the particle velocity and the speed of light which is small in the case of relativistic particles. In the case of 100 bins per variable the maximum slippage x_s in x between two time step is therefore usually one to two bins. In order to identify whether two bunches detected at timestep t and $t+1$ are the same the merging only needs to check whether $x_i(t) - x_s \leq x_i(t+1) \leq x_i(t) + x_s$. This has shown in practice to be a reliable approach for identifying corresponding bunches since it does not make any hard assumptions on the minimal temporal resolution of the data. In terms of the temporal resolution of the data the analysis here only assumes that the assumption $A2$ is true (see Section 5.6.1).

By using $x_i(t)$ to correlate different bunches the algorithm furthermore assumes that $A1$: *"Within each ROI in x/y space only one high quality bunch of interest exists at a time."* is true (see Section 5.6.1). Since particle bunches accelerate and decelerate over time their location in $x/y/px$ space within the corresponding ROI will change over time. In practice $A1$ has shown to be a reasonable assumption. In cases where several potential bunches exist within the same peak in $x/y/px$ space, the merging procedure combines the two bunches. As explained below, in this case the algorithm will make an automatic, implicit decision and retrieve the bunch that defines the highest density feature for the longest period of time which usually is the bunch that shows the highest acceleration.

In order to define a single consolidated description for the different bunches the merging function needs to combine the information from the different times steps, i.e., the lists of particle IDs defining a bunch. The goal here is to define for each bunch: (i) a list of candidate particles describing which particles potentially belong to the bunch, and (ii) identify a list of reference particles that have a high probability of belonging to the bunch. In order to define the list of candidate particles the algorithm simply merges the particle ID lists from the different timesteps that define the same bunch. In this process the merging function counts for each particle ID how often the according particle was found as being part of the bunch. For example, when a particle was found at 10 different timesteps as being part of a particular bunch then this particle was assigned a count of 10. This so defined function $l_b(id_i)$ defines for each bunch b a discrete frequency measure also referred to as bunch frequency. The bunch frequency indicates the likelihood of a particle with $ID = id_i$ of being part of a particular bunch, i.e., the more frequent a particle was detected as being part of a bunch the more likely it is that this is in fact the case. Based on l_b

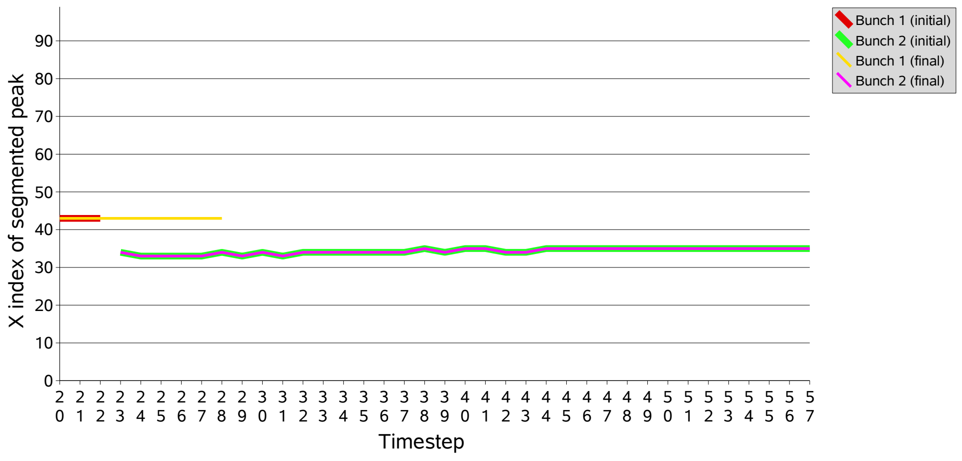


Figure 5.30: Overview of the merging process. In the example dataset bunches are initially segmented at the locations indicated by the thicker lines. In the merging process the algorithm then detects two main bunches shown in red and green. In the forward-tracing step additional segmentations are carried out for the first bunch. The thinner yellow and magenta lines indicate the final results after merging for the two bunches. One can see that in case of the example dataset additional bunch segmentations are only carried out in the forward-tracing of the first bunch.

the algorithm then identifies a set of reference particles for a particular bunch b by selecting the particles with the highest bunch frequency values l_b. More details later in Section 5.6.6.2.

Having defined how the algorithm solves the correspondency and combination problem the initial merging can be computed. The merging function iterates through the results from the initial timestep analysis and check for the bunch detected at the current timestep t whether it corresponds to the previous bunch detected at timestep $t-1$. If the two bunches are the same then their particle ID-lists are merged and l_b is updated accordingly. If the two bunches do not correspond then the number of detected bunches is increased, a new list of candidate particles is created for the new bunch as well as a new bunch frequency function l_{b+1}. The thick data lines in the plot shown in Figure 5.30, illustrate the results from the initial merging for the example dataset.

The initial timestep analysis detected only the single most prominent bunch at each timestep. To account for the fact that several bunches may coexist at the same time (see Section 5.6.1, F7), the algorithm first traces the detected bunches forward and afterwards also backward in time in order to complete the information for each bunches. In the following the last timestep at which the algorithm found a bunch b is referred to as $t_{last}(b)$. Starting from $(t_{last}(b)+1)$ the forward-tracing checks if a maximum of the function p_{max} that can be segmented exists at approximately the same location ($\pm x_s$) where the bunch b was previously found (i.e., $x_b(t_{last}(b))$). If this is

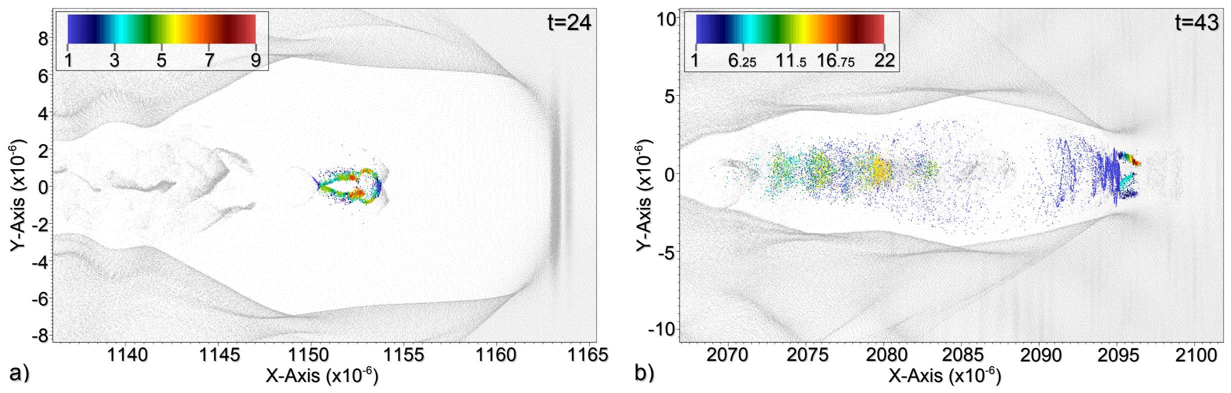


Figure 5.31: a) Bunch frequency l_1 of the first bunch shown at its peak px momentum at timestep $t = 24$. b) Bunch frequency l_2 of the second bunch shown at its peak px momentum at timestep $t = 43$. The candidate particles of the according bunch are colored according to the associated bunch frequency values indicating how often a particle was found as being part of the bunch. All particles not detected as candidates for the according bunch are shown in gray in both figures. Note, here only a subset of the complete simulation window containing all candidate particles of the displayed bunch is shown. One can see that in both cases the particles with the highest l_b values are highly localized defining the "core" of each bunch.

the case then the algorithms performs the bunch segmentation step for the identified location and merge the retrieved information with the current description of the bunch b, i.e., merging function updates the list of candidate particles, t_{last}, and l_b accordingly, and continue with the forward-tracing. If there is no maximum that can be segmented at the according location then the forward-tracing is terminated for the current bunch b and the forward-tracing function switches to the next bunch $(b+1)$. In the forward-tracing the algorithm may also detect that two previously separated bunches are actually the same. This may be the case when two bunches (e.g. b_1, b_3) correspond via their principle location but the according bunch was previously not segmented at all timesteps, e.g., the bunch was only segmented at timestep t and $t+2$. If the forward-tracing now detects that the same bunch also exists at timestep $t+1$ then it has closed the temporal gap in the description of the bunch indicating that the previously separated bunches b_1 and b_3 are actually the same. In this case the descriptions of these bunches are merged after closing the temporal gap and the forward-tracing is continued.

Similar to the forward-tracing the algorithm now also has to trace the different bunches backward in time. Note, in contrast to the forward-tracing no checks for whether two previously separated bunches are actually the same are necessary in the backward-tracing. All these cases have already been resolved by the forward-tracing.

Figure 5.30 illustrates the merging process for the example dataset. In this case the beam path analysis detected two different bunches. Initially the first bunch was only detected at timesteps $t = [20, 22]$. The forward-tracing then found that the first bunch also exists at timesteps $t = [23, 28]$ and added the according information to the initial description of the bunch. The backward-tracing did not result in any additional information in this particular case. Note, when performing additional bunch segmentations in the forward and backward-tracing the algorithm only needs to re-execute the actual bunch segmentation (see Section 5.6.4.2) but not the expensive data preparation step (see Section 5.6.4.1) of the timestep analysis.

The plot shown in Figure 5.30 provides a first rough overview of the lifetime of the detected bunches. The different bunch frequency functions l_b shown in Figure 5.31 then describe a first approximated classification of the different bunches. One can see that in both cases all particles with very high l_b values — $l_1 \geq 7$ (with $max(l_1) = 9$) and $l_2 \geq 19$ (with $max(l_1) = 22$) — are each located within a confined region in physical space. These are also the particles the algorithm will define later as references for the first and second bunch respectively. In case of the second bunch (see Figure 5.31b), one also sees a larger number of particle with medium bunch frequency values $l_2 \approx 11$ appearing further in the back at a larger distance to the main bunch. This is due to the fact that these particles form a high-density feature in $x/y/px$ space during early timesteps $t \approx 23$. This bunch appears, however, at very low energies and decelerates quickly whereas the second main bunch appears at high energies and exists for a longer period of time. The particles with highest l_2 values are therefore all located within the main bunch of interest and are condensed in physical space.

5.6.6 Particle Path Analysis

The goal of the particle path analysis is to compute an accurate description of the detected particle bunches based on their complete temporal history. After the merging process the number of particles the analysis needs to consider has greatly been reduced to a set of candidate particles for each bunch. As illustrated in Figure 5.32 the particle path analysis consists of the following steps: (i) particle tracing (see Section 5.6.6.1), (ii) reference path analysis (see Section 5.6.6.2), and (iii) path distance computation (see Section 5.6.6.3).

5.6.6.1 Particle Tracing

As the first step of the particle path analysis the particle tracing computes the complete temporal paths of all candidate particles of the current bunch. One, therefore, needs to locate the candidate particles at every timestep of the simulation. Using FastBit the algorithm can access the

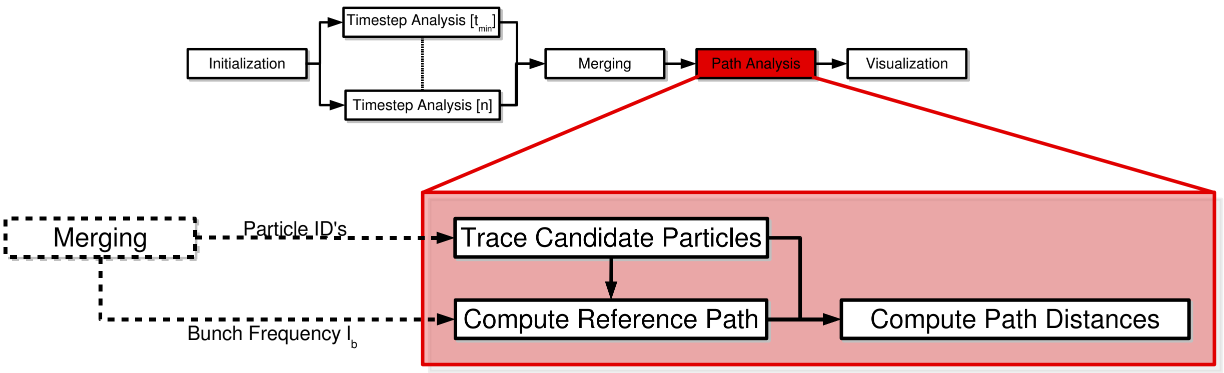


Figure 5.32: **Overview of the particle path analysis:** This step is performed independently for each identified bunch. First, all candidate particles of a bunch are traced over the complete timeseries. Based on the bunch frequency function l_b the algorithm defines a set of reference particles for the bunch and derive from their paths a single reference path describing the temporal evolution of the bunch. Based on the reference path the algorithm first computes the different temporal phases of the according bunch, defining, e.g., when the bunch was formed or accelerated. Afterwards, the path distance computation defines for each candidate particle the distance of its path to the reference path. The computed path distance fields allow the user to accurately describe the particle bunch.

relevant data efficiently by executing an according equality query of the form $ID = id_1 || ID = id_2 || .... || ID = id_n$ at all time steps of the data set and then load the associated data.

The number of candidate particles per bunch largely depends on the resolution of the raw data, i.e., the number of particles per timestep. In practice the number of candidate particles is usually in the order of the magnitude of a couple of thousand up to several ten-thousand particles. The amount of data that need to be accessed in order to define the complete temporal history of all candidate particles is therefore in practice small and can efficiently be managed even on a regular desktop computer. The same basic method for tracing particles was also employed in earlier work to enable fast visual exploration of the data [158].

5.6.6.2 Reference Path Analysis

To define the distance of a particle to the current bunch b the algorithm uses a reference path to describe the temporal evolution of the bunch. To compute the reference path the algorithm therefore needs to first identify a set of reference particles that characterize the bunch.

As mentioned earlier, the bunch frequency function l_b (computed during the merging) can be used effectively to identify a group of particles that define the “core” of the bunch. Particles with very high l_b values define a group of particles that were consistently detected as being part of

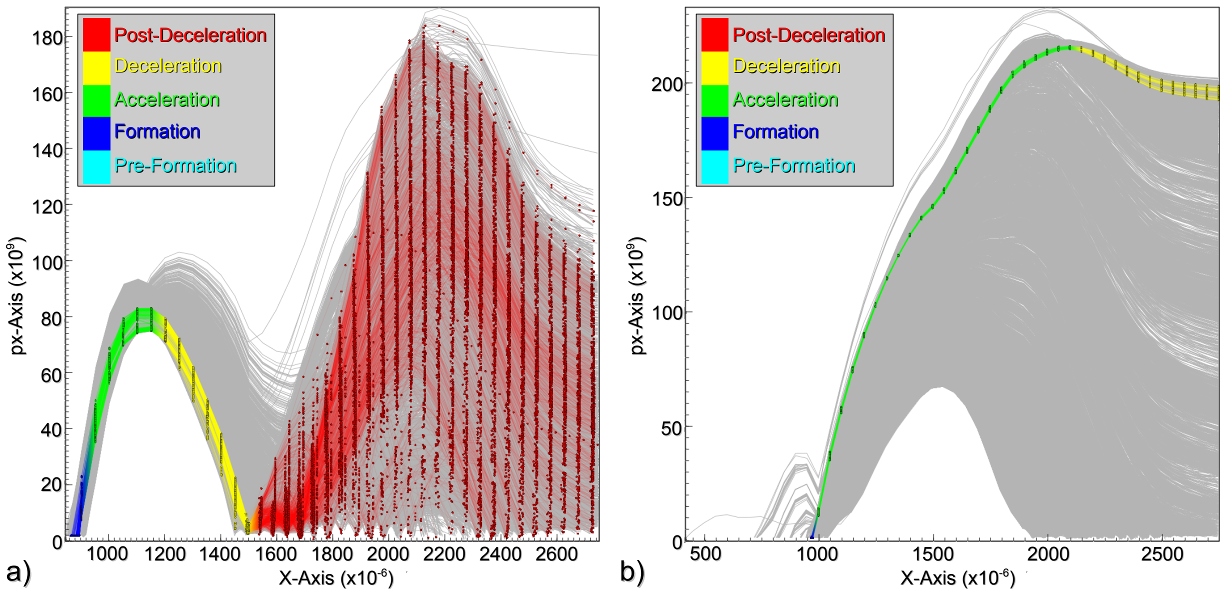


Figure 5.33: a) Temporal phases of the first bunch. b) Temporal phases of the second bunch. Both figures show the paths of all candidate particles of the according bunch in gray. The reference particles and their paths are shown in addition colored according to the different beam-phases. The second bunch (b) does not show a post-deceleration phase because the simulation terminated before the bunch had completed deceleration.

the bunch b. Furthermore, the bunch segmentation is always centered around the highest density feature within the according peak in $x/y/px$ space, i.e., the density core of the bunch. Particles with very high l_b values therefore define a compact group of particles centered at the density core of the bunch (see also Section 5.6.5 and Figure 5.31). Due to potential over-segmentation at single timesteps during the initial timestep analysis one may in practice only find a few particles with the maximum l_b value for a bunch. The algorithm can therefore not just simply select only those particles with the maximum l_b value but needs to ensure that it selects a sufficient amount of particles with high l_b values to describe the bunch accurately. The reference path analysis, therefore, first computes the histogram for l_b — defining how many particles were found at each discrete bunch frequency level — and then detects the local maximum m of the histogram with the highest l_b value. All particles with $l_b \geq m$ are then chosen as reference for the current bunch b. In case of the example dataset the reference levels for the two detected bunches are $l_1 = 7$ (with $max(l_1) = 9$) and $l_2 = 19$ (with $max(l_2) = 22$), respectively.

From the temporal paths of the reference particles the algorithm then computes a single reference path describing the temporal behavior of the current bunch b by computing the average in x, y, z, px, py, pz of the reference particles at each timestep t. Based on the reference path different temporal phases of the bunch are defined as follows:

- **Pre-Formation:** Early time frame during which less than 80% of the reference particles are present, i.e., the bunch is not well characterized yet.
- **Formation:** Early time frame during which more than 80% of the reference particles are present but the momentum in x direction is still low, i.e., $px < 10^{10} ms^{-1}$.
- **Acceleration:** Time frame directly after the beam formation phase during which the beam is constantly accelerated until it reaches its peak energy, defined via px. Note this may not be the global maximum in px for the particles of the bunch. As illustrated in Figure 5.33a, particles may undergo secondary phases of acceleration after the beam of interest has lost its coherency.
- **Deceleration:** Timeframe directly after the beam has reached its peak px momentum and is constantly decelerating.
- **Post-Deceleration:** This phase includes all timesteps after the beam has completed its deceleration phase. The beam has lost its coherency so that its behavior is undefined during this phase. For example, some of the particles may become trapped in secondary periods of the wave and undergo a secondary phase of acceleration (see Figure 5.33a), while other particles may leave the simulation window and are no longer traceable.

Figure 5.33 illustrates the different phases of the two bunches detected in the example dataset. The fact that no pre-formation phase can be seen along the reference paths of the two bunches indicates that most reference particles ($> 80\%$) enter the simulation window at the same timestep. When comparing the paths of the reference particles of the two bunches shown in Figure 5.33a and b, one can see that different bunches may show a different acceleration behavior. Information on the different phases of a bunch provides the user with valuable information on which timesteps are of interest for visualization and other types of analysis. The algorithm also uses this information in the later path distance computation in order to be able to accurately classify the detected particle bunches (see Section 5.6.6.3).

5.6.6.3 Path Distance Computation

The path distance computation is the final step of the main analysis with the goal to define for each candidate particle its distance to the bunch, i.e, the distance of the particle's path to the reference path of the bunch. Based on the path distance, accurate classification of a particle bunch becomes possible. The path distance function should fulfill the following basic requirements:

- It should be smooth and allow the user to effectively define the bunch with respect to pre-knowledge and current analysis requirements.
- The function should be intuitive and physically meaningful.
- Function values should have a physically meaningful scale.
- Function values, and in this way classifications of different bunches, should be comparable.

In order to avoid any non intuitive normalization, achieve comparability, and ensure that function values are at a physically meaningful scale the algorithm computes not a single but two independent path distance functions. In the case of LWFA particle data exist essentially two different main data spaces: (i) physical space with the dimensions x, y, z; and (ii) momentum space with the dimensions px, py, pz. In these two spaces one can directly define the standard Euclidean distance with no need for normalization. The Euclidean distance in physical space between two particles with index i and j at time t is defined as:

$$d_s(i,j,t) = \sqrt[2]{(x(i,t)-x(j,t))^2 + (y(i,t)-y(j,t))^2 + (z(i,t)-z(j,t))^2} \tag{5.3}$$

With $x(i,t)$, $y(i,t)$ and $z(i,t)$ being the location of the particle with index i at time t in x, y, and z respectively. The according distance in momentum space is defined as:

$$d_m(i,j,t) = \sqrt[2]{(px(i,t)-px(j,t))^2 + (py(i,t)-py(j,t))^2 + (pz(i,t)-pz(j,t))^2} \tag{5.4}$$

With $px(i,t)$, $py(i,t)$ and $pz(i,t)$ being the momentum of the particle with index i at time t in x, y, and z direction respectively. In the case of a 2D simulation the terms referring to z and pz are ignored.

The distance between the temporal path of two particles with index i and j in physical and momentum space is defined as the average distance along their path, i.e.:

$$d_s(i,j) = \frac{\sum_{t=t_k}^{t_l} d_s(i,j,t)}{n_t} \tag{5.5}$$

$$d_m(i,j) = \frac{\sum_{t=t_k}^{t_l} d_m(i,j,t)}{n_t} \tag{5.6}$$

With $n_t = t_l - t_k + 1$ (with $t_l > t_k$) being the number of timesteps considered in the computation of the path distance. Figure 5.34 illustrates the computation of the distance between the path of two particles.

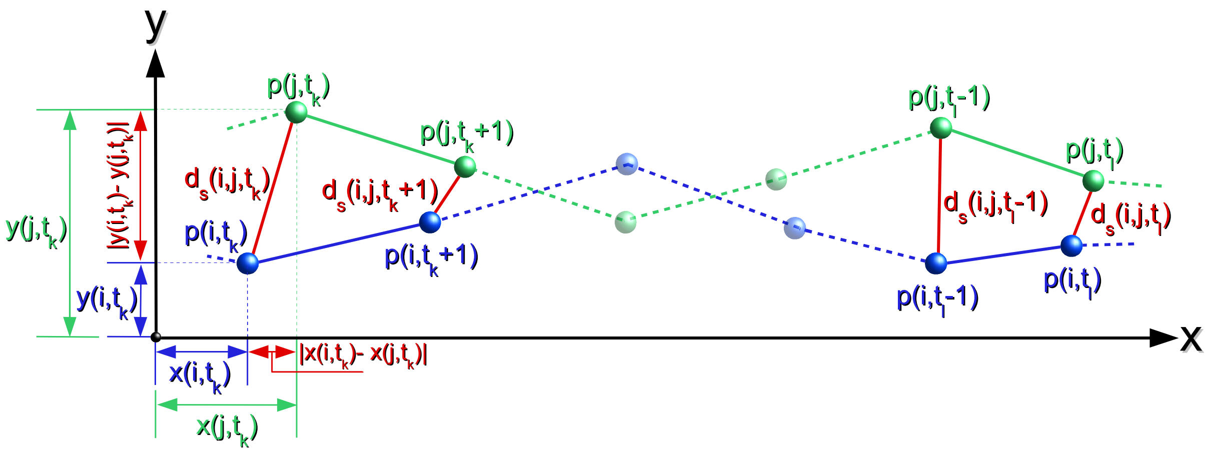


Figure 5.34: Illustration of the path distance computation for two particles with index i and j in 2D physical space x/y. The distance in momentum space is defined similarly.

The distance of a particle to a bunch is defined as the distance of the particle's path to the reference path of the beam. During the pre-formation and formation phase a beam is not well defined yet. During the deceleration phase a beam looses its coherency and has fallen apart during the post-deceleration phase. The only timeframe during which a beam is well defined is during the acceleration phase. The path distance computation is therefore restricted to the acceleration time frame of the bunch by defining t_k and t_l accordingly. The algorithm then computes for each candidate particle the respective distances d_s and d_m to the reference path of the current bunch. These distances fulfill all basic requirements mentioned above; they are in a physically meaningful scale —d_s in $meter$, d_m in ms^{-1}—, intuitive, physically meaningful and smooth. Path distances from different bunches are also comparable, i.e., a basic distance of, e.g., $d_s = 10^{-6}m$ or $d_m = 10^{11}ms^{-1}$, have the same basic meaning for different bunches. In order to define which particles actually belong to a bunch the user then needs to define according thresholds in d_s and d_m to define the maximally allowed distance in physical and momentum space to the beam. In order to compare different bunches a user may use the same threshold values for different bunches.

While this thesis describes the use of the average distance along particle paths to define a bunch, other distance functions can easily be defined to describe other beam characteristics. Other distance functions of interest may, e.g., be the minimum distance along paths or also just the distance at the timestep where a bunch has reached its peak momentum in x-direction (px).

Figure 5.35 and 5.36 provide an overview of the two bunch classification functions d_s and d_m for the two bunches detected in the example dataset. One can see that in both cases the analysis was able to identify the bunch properly and that d_s and d_m define in each case the same principle

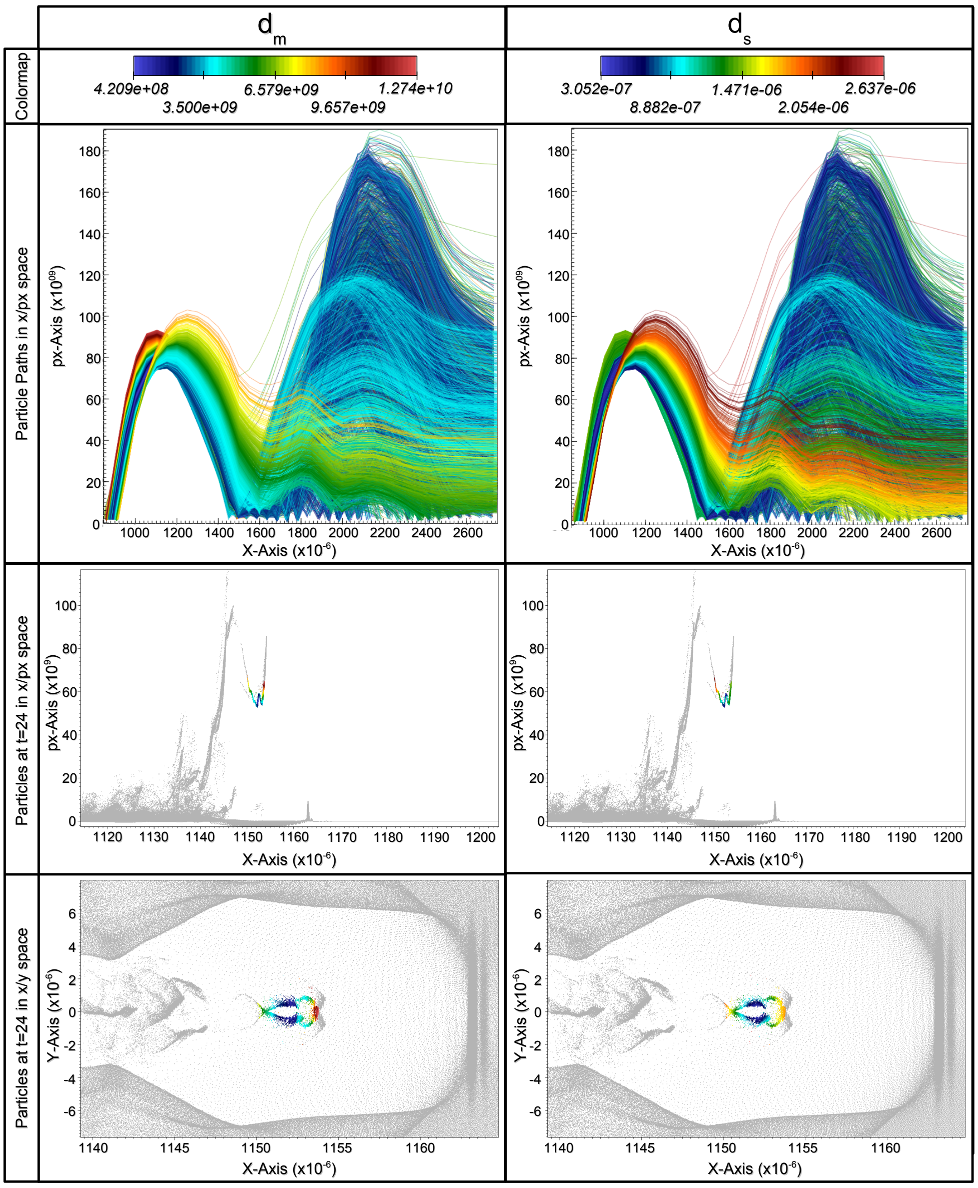


Figure 5.35: Overview of the path distance fields for the first bunch detected in the example 2D dataset illustrating that d_m and d_s accurately classify the bunch. Note, here all candidate particles are shown, i.e., no thresholds in d_s or d_m have been applied in the images. The left column of the table shows the distance in momentum space d_m and the right column the distance in physical space d_s. Color indicates d_m and d_s respectively. Third row: Temporal paths in x/px space of all candidate particles. Fourth row: Particles in x/px space. Fifth row: Particles in x/y space. The particle images indicate the state of the bunch at timestep $t = 24$ when the bunch has reached its peak momentum in px. Note, the bottom images show only a subset of the simulation window containing all detected candidate particles of the bunch. One can see that both distance functions define the bunch well. Results after thresholding are presented later in Section 5.6.7 (see Figure 5.38(C1)).

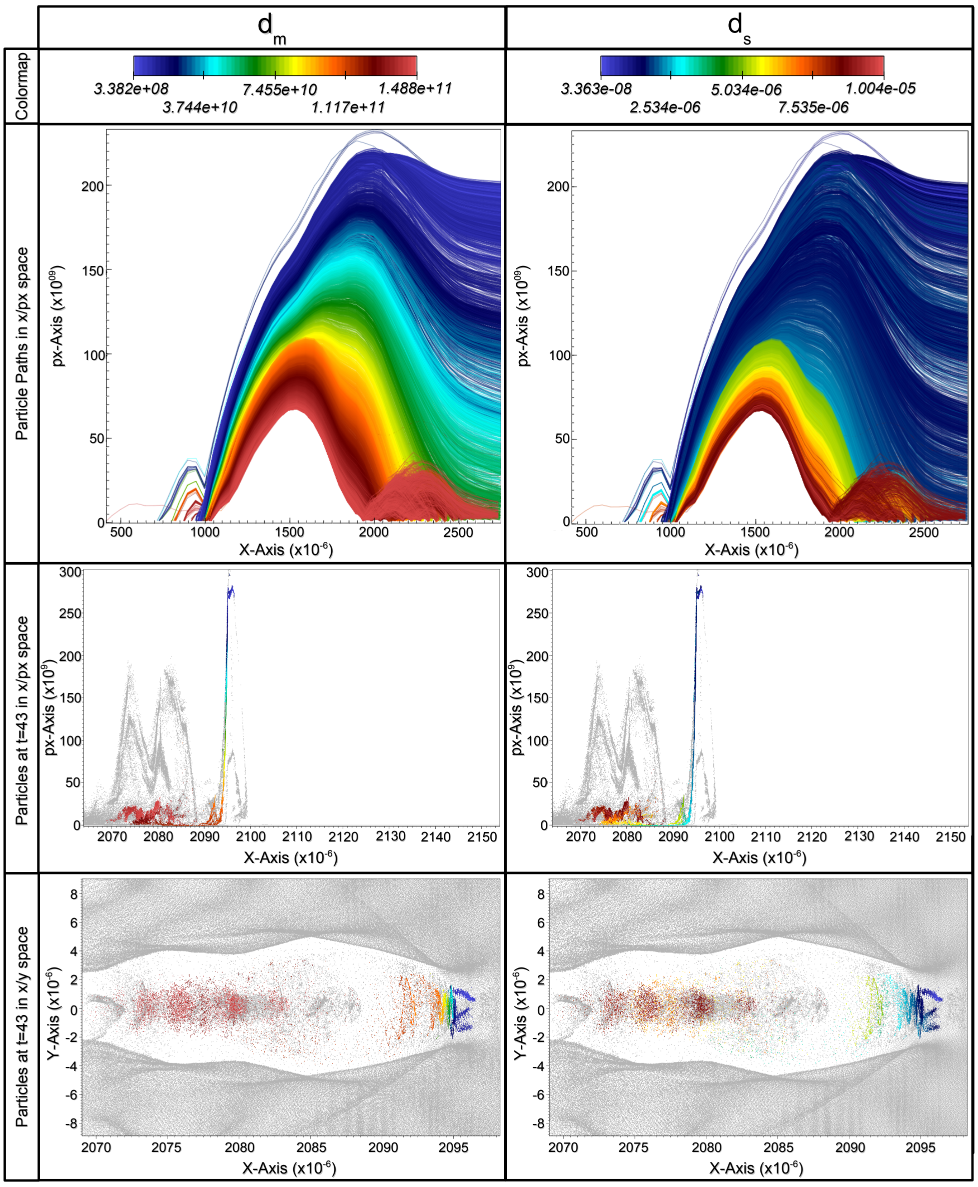


Figure 5.36: Overview of the path distance fields for the second bunch detected in the example 2D dataset illustrating that d_m and d_s accurately classify the bunch. Note, here all candidate particles are shown, i.e., no threshold in d_s or d_m have been applied in the images. The left column of the table shows the distance in momentum space d_m and the right column the distance in physical space d_s. Color indicates d_m and d_s respectively. Third row: Temporal paths in x/px space of all candidate particles. Fourth row: Particles in x/px space. Fifth row: Particles in x/y space. The particle images indicate the state of the bunch at timestep $t = 43$ when the bunch has reached its peak momentum in px. Note, the bottom images show only a subset of the simulation window containing all detected candidate particles of the bunch. One can see that both distance functions define the bunch well. Results after thresholding are presented later in Section 5.6.7 (see Figure 5.38(C2)).

bunch. As expected, there are also outliers in physical as well as momentum space, i.e., particles that show a similar acceleration behavior as the bunch but are distant in physical space as well as particle that stay close to the bunch in physical space but are accelerated differently. By applying according thresholds in d_s an d_m one can accurately define each of the two bunches.

5.6.7 Validation

This section presents results of the beam path analysis for a variety of different datasets described in Table 5.2. Figure 5.37 to 5.39 give an overview of the results our method achieves on theses different datasets. For each bunch the resulting phase space diagram (x/px) is shown at the timestep when it has reached its peak momentum in px. These plots show all candidate particles in blue and all particles that satisfy the default selection condition $(d_s < 2 * 10^{-6})\&\&(d_m < 10^{10})$ in red. This condition ensures only particles that are close to the bunch in momentum as well as physical space are selected. Depending on the current requirements a researcher may in practice choose lower or higher thresholds for d_s and d_m. As the different plots of d_s and d_m illustrate, both functions are smooth and enable the user to accurately define each bunch.

Dataset	**Type**	**Size per timestep in MB≈**	**Total size in MB≈**	**Number of timesteps**	**Particles per timestep≈**
A	2D	35	1,320	38	405,000
B	2D	128	4,606	36	1,600,000
C	2D	190	11,034	58	2,400,000
D	2D	62	13,990	226	610,000
E	3D	23,999	623,964	26	229,850,000
F	3D	7,091	212,733	30	90,790,000

Table 5.2: Description of the simulation datasets used for validation.

Traditionally thresholding in px is used to identify high energy particles and the beam(s) of interest. In order to identify proper thresholds and timestep, a researcher investigates movies of a variety of plots, a complex and time-consuming process. In many cases a single threshold may not be sufficient to isolate the beam particles of interest, e.g., when two high energy bunches exist at the same time as, e.g., in dataset C. The main deficiency of manual thresholding are that it is arbitrary, time-consuming, and requires manual inspection of the dataset.

As summarized in Table 5.2, the datasets used in this validation are of varying temporal as well as spatial resolution, ranging from 26 to 226 dumped timesteps and $\approx 405,000$ to

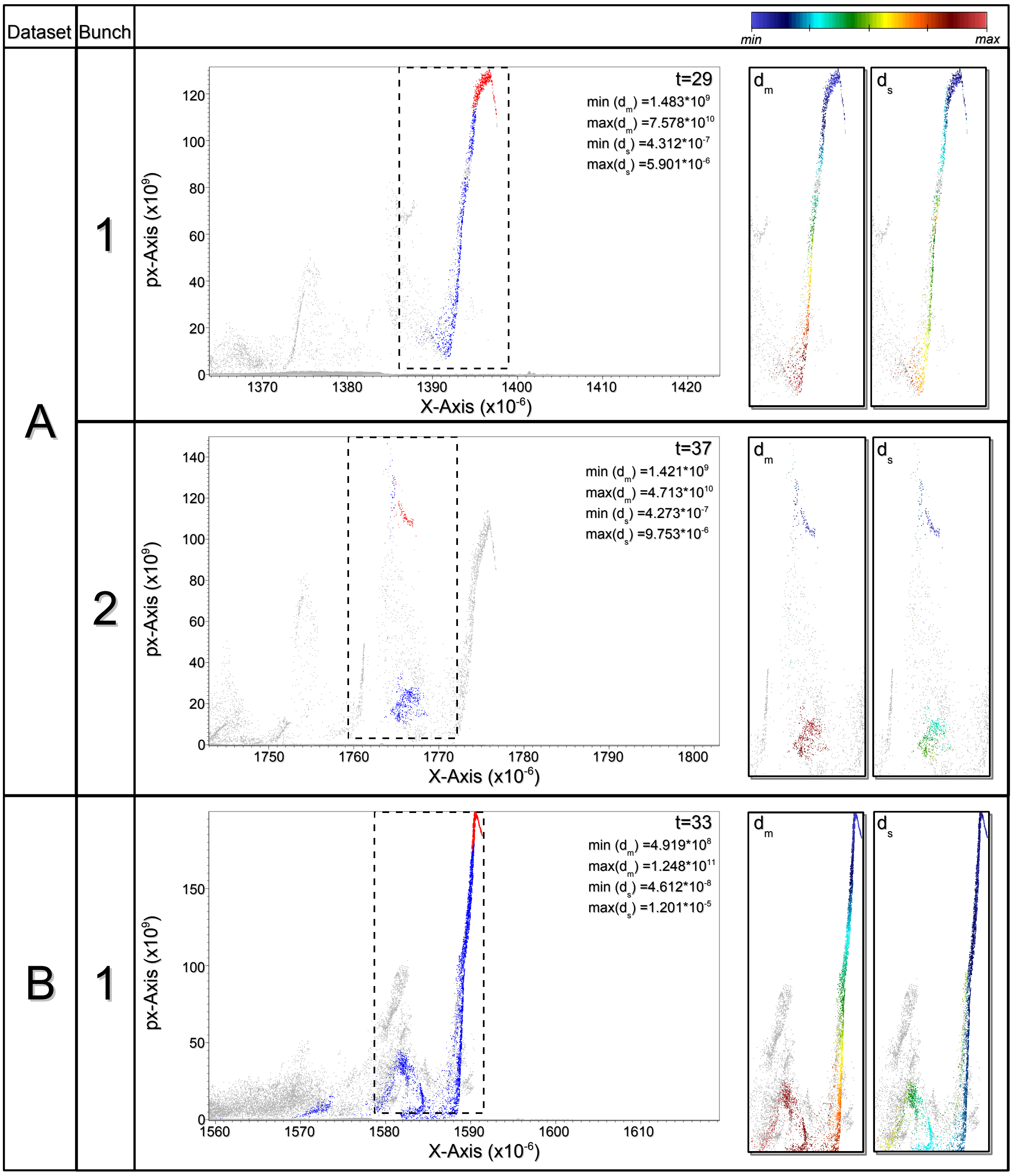


Figure 5.37: Overview of the analysis results for dataset A and B illustrating that the algorithm is able to accurately detect the relevant bunches. In the main figure all detected candidate particles are shown in blue and all particles that satisfy the condition $(d_s < 2 * 10^{-6}) \&\& (d_m < 10^{10})$ are shown in red. The applied condition ensures that only particles close to the bunch are selected. The cutout plots show the distance functions d_m and d_s respectively. Each bunch is shown at the timestep when it has reached its peak px level.

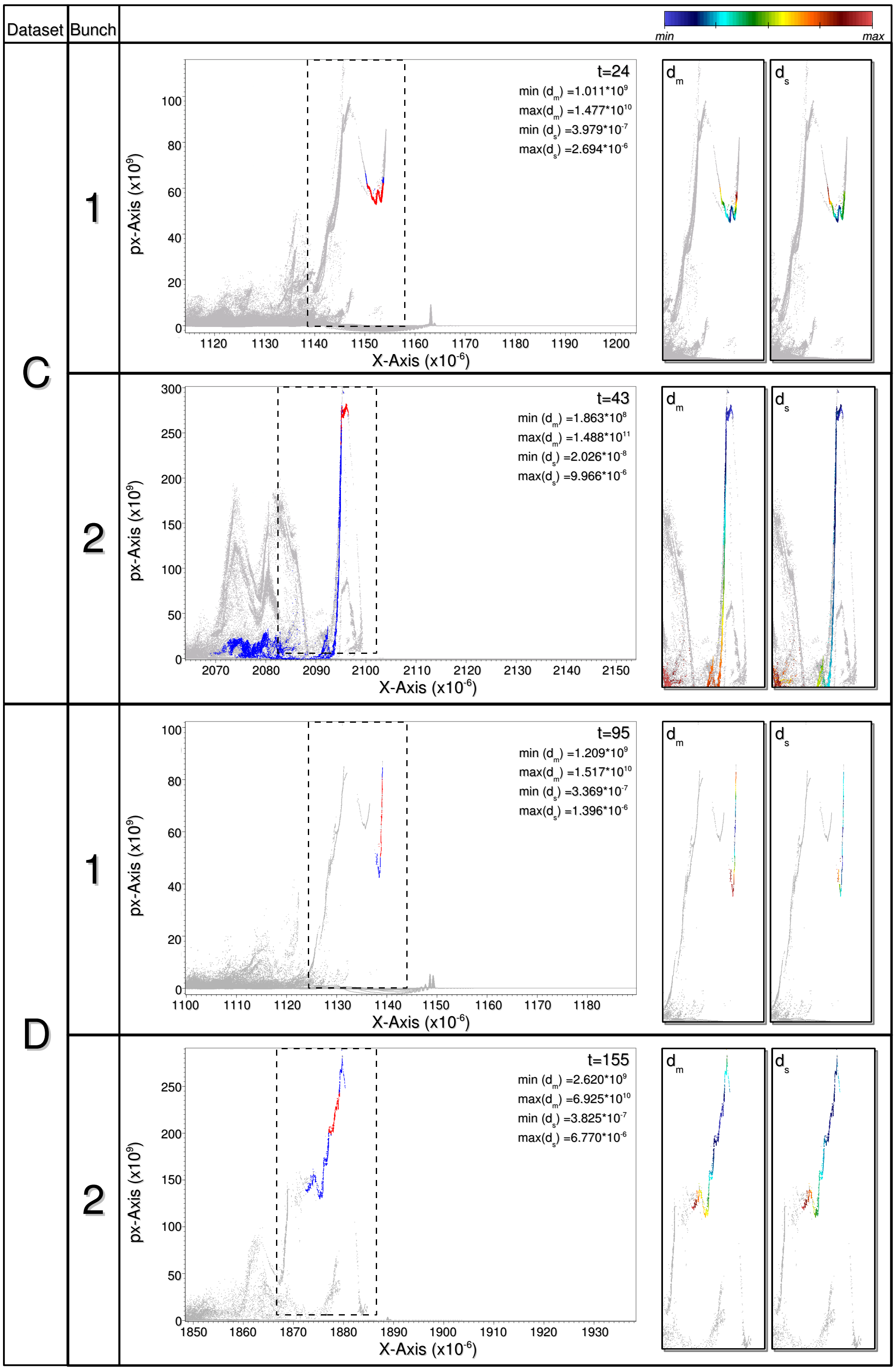


Figure 5.38: Overview of the analysis results for dataset C and D illustrating that the algorithm is able to accurately detect the relevant bunches (see also Figure 5.37).

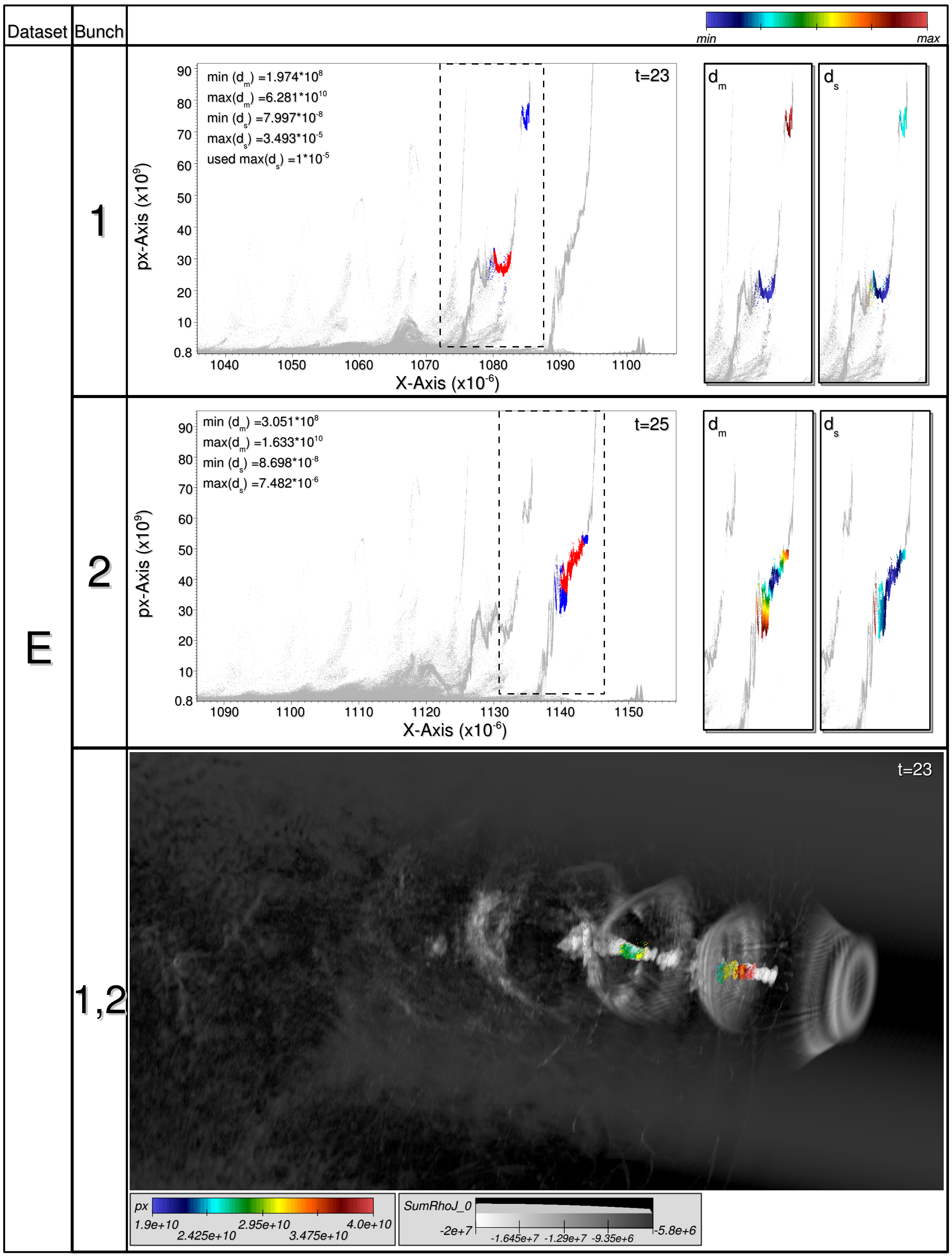


Figure 5.39: Overview of the analysis results for the large 3D dataset E illustrating that the algorithm is able to accurately detect the relevant bunches (see also Figure 5.37). The bottom image shows a volume rendering of the plasma density (gray) and the two selected bunches colored according to px at timestep $t = 23$ illustrating the location of the two bunches within the plasma wave.

$\approx 229,850,000$ particles per timestep. Dataset F was produced using an older version of the simulation code and does not contain unique particle IDs. In the context of the beam path analysis dataset F is only used to evaluate the performance of the computation of 3D histograms and 3D bin-queries for which IDs are irrelevant.

As Figure 5.37(A) shows, the beam path analysis algorithm reliably detects particle bunches even in datasets with low spatial resolution. Dataset B and C are two 2D dataset with medium spatial resolution. Dataset D is also a 2D dataset but with high temporal resolution. The results for datasets B-C are shown in Figure 5.37(B) and 5.38(C,D) respectively. Dataset E is a massive 3D dataset with $\approx 229,850,000$ particles per timestep. Due to the large amount of data that needs to be stored just for a single timestep only few timesteps are written to file. The sheer size of this dataset as well as its low temporal resolution make detection of particle bunches a challenging tasks. As shown in Figure 5.39, even in this most challenging case the method is able to reliably detect the two main bunches of interest.

5.6.8 Performance Evaluation

This section presents an analysis of the performance of the beam path analysis algorithm and its different parts. Section 5.6.8.1 describes the performance of the 3D histogram computation used in the timestep analysis (see Section 5.6.4). Afterwards, Section 5.6.8.2 characterize the performance of 3D bin queries in , i.e., how much time does it take to identify the particles located within a set of selected bins of a 3D histogram. Both, the 3D histogram computation as well as 3D bin queries, are crucial parts of the initial timestep analysis (see Section 5.6.4) aimed at identifying particle bunches of interest at a particular timestep. The performance tests are performed based on a representative timestep of the medium sized 2D dataset (C) and a 3D dataset (F) (see Table 5.2) containing a condensed particle bunch of interest. Section 5.6.8.3 then analyzes the serial performance of the complete analysis algorithm using the datasets described in Table 5.2. The serial analysis tests are aimed at characterizing the overall performance of the beam path analysis algorithm as well as the performance of the different parts of the pipeline. A detailed analysis of the performance of the particle tracing using FastBit was already presented earlier in Section 5.4.5.

For all performance tests described in Sections 5.6.8.1 - 5.6.8.2 a workstation equipped with two $2GHz$ dual core AMD Opteron$^{\mathrm{TM}}$270 processor, $8GB$ running SuSE Linux was used. In the serial performance the tests only one of the processing cores is actually used by the program.

5.6.8.1 Computing 3D Histograms

This performance test is aimed at characterizing the performance of the computation of 3D conditional histograms. The goal is to characterize the speed-up achieved through the use of FastBit for computing the histograms as well as the overhead for computing the bit vectors in addition to the bin counts. This performance study, therefore, compares the performance of the following different implementations:

- **H1: Bitvectors:** This is the implementation used in the analysis code. FastBit first evaluates the condition and then computes for each non empty bin a bit vector indicating which particles belong to the according bin. From these bit vectors the counts of the 3D histogram are then derived.
- **H2: FastBit:** This implementation uses FastBit to first evaluate the condition of the histogram. Afterwards the counts of the histogram are computed based on the selected particles only.
- **H3: Sequential:** This variant does not make use of FastBit as acceleration data structure. The implementation performs a sequential scan through all particles and checks for each particle if it satisfies the given condition and updates the counts of the 3D histogram accordingly.

For each implementation a series of 3D histograms in *x*, *y*, and *px* are computed with varying numbers of bins per variable. As in the timestep analysis the test-code uses the condition

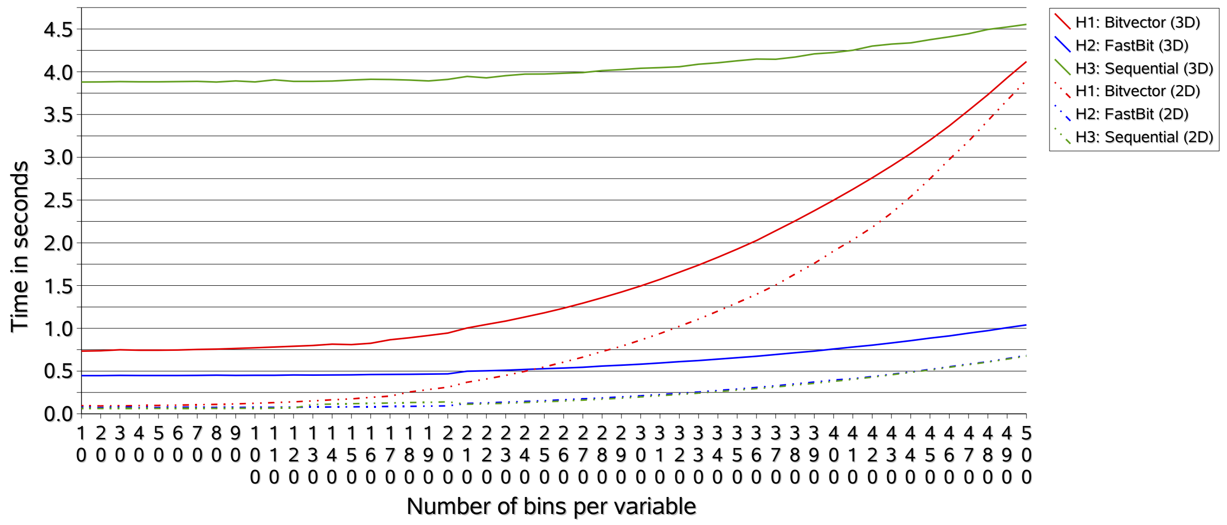


Figure 5.40: Timings for serial computation of conditional 3D histograms.

$(px > 10^{10}) ms^{-1}$. All histograms are computed over the complete data range of the according variables. The test-code repeats each measurement twenty times while here the average timings are reported.

Figure 5.40 shows the performance of the different implementations for the example timesteps of dataset C and F. For the smaller 2D dataset H2 and H3 show similar performance. For the larger 3D dataset H2 then shows a much better performance than H3. This is expected since the relative cost for evaluating the condition using FastBit compared to the additional cost for also traversing the particles that do not satisfy the condition will decrease with increasing file size. For less than ≈ 150 bins per variable H1 is in general slower than H2 by a roughly constant factor of ≈ 1.3 to ≈ 2. For larger numbers of bins the performance of H1 decreases faster than for H2. This is due to the fact that besides the counts of the histogram the test function H1 also has to maintain a secondary data structure (the bit vectors) which requires random access to memory. Updating this data structure will become increasingly expensive the larger it becomes. Note, the beam path analysis typically uses only ≈ 100 bins per variable where the cost of maintaining the bit vectors is modest. As the next section shows, the use of bit vectors significantly improves the performance of 3D bin queries and the additional time needed to compute and maintain the bit vectors is well spent.

5.6.8.2 Evaluating 3D Bin Queries

As part of the bunch segmentation process the beam path analysis algorithm needs to identify which particles are located within selected bins of a 3D histogram. As described earlier, the algorithm uses a set of bit vectors — computed during the 3D histogram computation — in order to be able to access the data of these particles efficiently. In case that this inverse mapping from the 3D histogram back to the original data space is not available then one will have to evaluate a corresponding query in order to decide which particles are associated with the selected bins. For each bin such a query takes the form of:

$$[(x_i \geq x)\&\&(x < x_{i+1})\&\&(y \geq y_i)\&\&(y < y_{i+1})\&\&(px \geq px_i)\&\&(px < px_{i+1})] \tag{5.7}$$

The parameters x_i, y_i, px_i and x_{i+1}, y_{i+1}, px_{i+1} indicate the lower and upper boundaries of the bin with index i respectively. In a typical segmentation not one, but several bins are selected so that many of these queries need to be combined via OR ($||$). For example, if 10 bins are selected then one has to evaluate a query consisting of 60 conditions combined via AND (&&) and OR ($||$).

The test-code used for the performance tests described in this section simulates the segmentation process by selecting the n most populated bins. The test-code then increases n to analyze the performance with increasing size of the selection and query complexity. This performance study compares the following different implementations:

- **Q1: Bitvectors:** This is the version used in the actual analysis code. The bit vectors of the selected bins are first merged and then FastBit is used to load the IDs of the selected particles.
- **Q2: FastBit:** This implementation uses FastBit to evaluate the segmentation query and then load the IDs of the selected particles only.
- **Q3: Sequential:** This implementation first performs a sequential scan through the data to evaluate the query. The test function first loads the data of all particles in x, y, and px and then checks for each particle whether it suffices the query. Since the different bin-queries are combined with OR the test function can stop this process for each particle as soon as the particle has been identified as being part of a selected bin. Furthermore, the selected bins are sorted in decreasing order with respect to their counts. If N is the total number of particles, M the number of selected particles, and B the number of selected bins then one needs to perform in the worst case: $(N - \frac{M}{2}) * (B * 6)$ compare operations to evaluate the query. Afterwards the test function again loads the IDs of the selected particles.

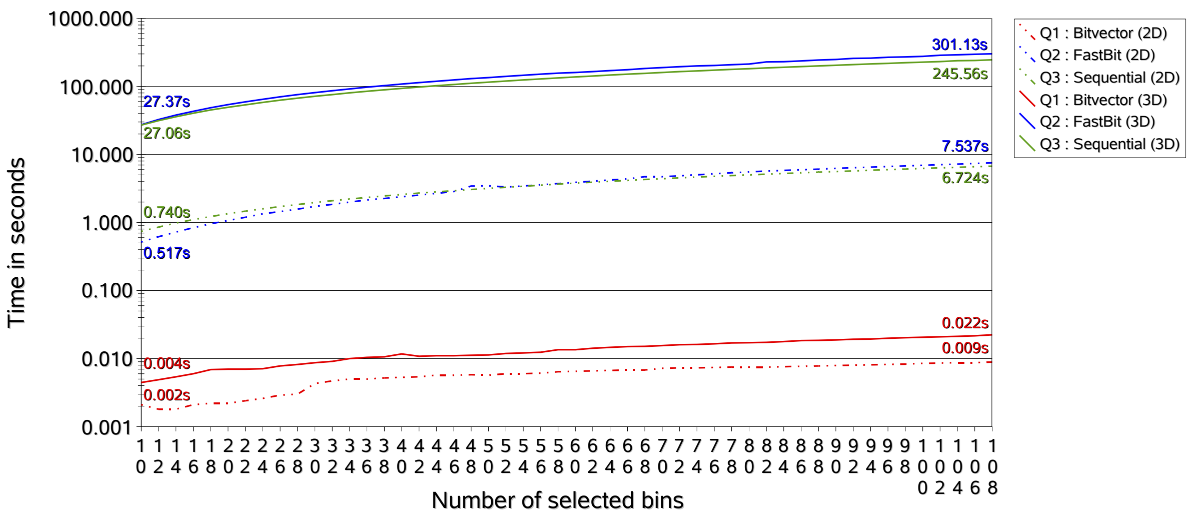


Figure 5.41: Timings for serial computation of 3D bin queries. Note the logarithmic scale on the y-axis showing the time in seconds. One can see that the data access using the bit vectors (Q1) is significantly faster than when having to evaluate a 3D bin query (Q2 and Q3).

Figure 5.41 shows the performance of the different implementations for the example timesteps of dataset C and F. The test-code repeated each measurement ten times while here the average timings are reported. Q2 and Q3 show a similar performance in the 2D case. For the larger 3D data file the sequential scan (Q3) performs better than the implementation using FastBit (Q2). To answer 3D bin queries of the form shown in Equation (5.6.8.2), FastBit invokes the indices on *x*, *y* and *px* separately. Currently, FastBit is not able to share intermediate results computed for different bins and therefore performs considerable amount of redundant work. Instead of optimizing this query evaluation procedure, having the 3D histogram computation returning bit vectors indicating which particle are associated with each bin has shown to be a more efficient approach. Compared to both, Q2 and Q3, the implementation used in the analysis-code (Q1) shows an outstanding performance in all cases. Even in the case of the relatively small 2D dataset when just selecting 10 bins one sees a speed-up of ≈ 258 compared to Q2 and a speed-up of ≈ 370 compared to Q3. When comparing Q1 to Q2 or Q3 in the case of the larger 3D dataset then one sees even higher speed-ups of ≈ 4500 when selecting only ten bins and $> 11,000$ when selecting more than 100 bins. The additional time needed to compute the bit vectors is justified by the large speed-up gained during the evaluation of segmentation results (and therefore also the merging).

5.6.8.3 Serial Performance of the Analysis Algorithm

The serial performance tests of the algorithm presented in this section were performed on a system equipped with eight $2GHz$ dual core AMD Opteron^{TM}Processor 870 with $8GB$ of memory per core running Ubuntu Linux. For the serial analysis only one core was used and the memory limit was set to $8GB$ while the peak memory usage of the analysis did not exceed $2.5GB$ in any case. The peak memory usage is reached in the case of dataset D due to its many timesteps and hence the large amount of data and information that is acquired in the initial timestep analysis.

Figure 5.42 shows the timings for the analysis of the datasets described in Table 5.2. Each measurement was repeated ten times while here the timing of the run with the median total analysis time is reported. One can see that even when run in serial, the beam analysis algorithm shows very good performance even for very large datasets. In order to give an overview of the complete performance of the algorithm, the minimum timepoint computation was executed for all datasets. As mentioned earlier, in practice a user often has already a good understanding of when the first particle beams are forming so that the minimum timepoint computation is often omitted.

The initialization step is where the algorithm first touch all necessary files and access meta-

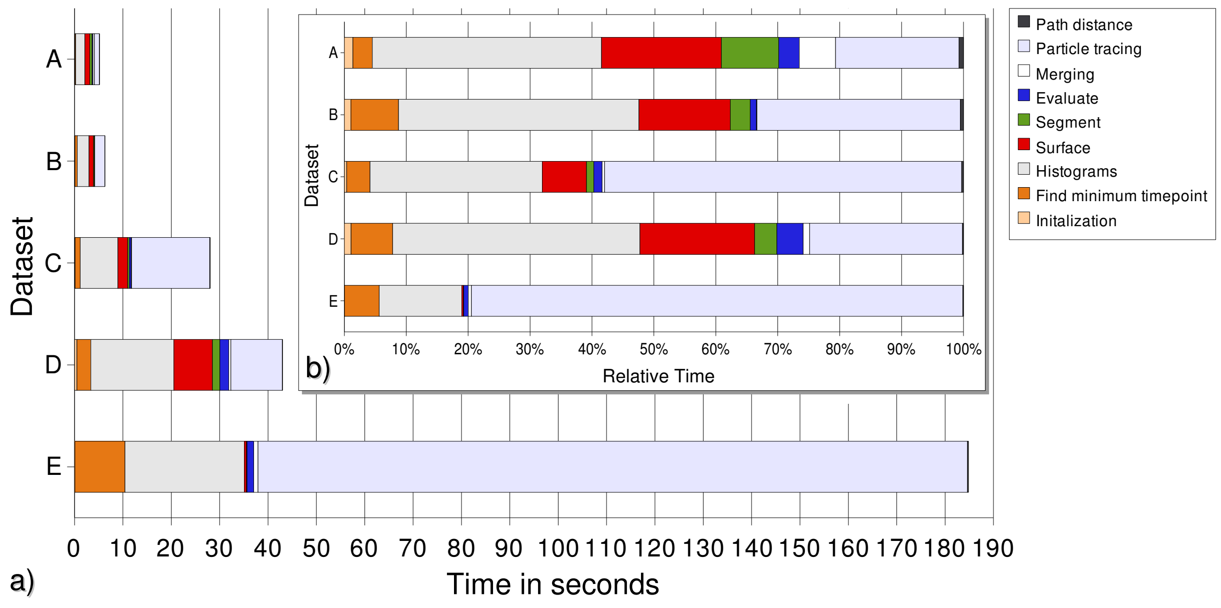


Figure 5.42: a) Absolute timings for the serial analysis of five different datasets using the beam path analysis method. The length of each horizontal bar represents the total time used for the complete analysis of the according dataset. b) Relative timings for the different steps of the analysis pipeline as percentage of the total analysis time. In both figures color is used to indicate the timings of different parts of the algorithm. In all cases the histogram computation and the particle tracing are the most expensive analysis steps. This is expected since these are the two main steps during which the raw data needs to be accessed. One can see that the algorithm shows good performance in all cases and scales well with increasing dataset size.

data. The initialization time largely depends on the time needed for the first disk access. Overall the 3D histogram computation and the particle tracing are the most expensive steps of the analysis pipeline. This is expected since these are the two main steps where the algorithm needs to access larger portions of the raw data. The particle tracing then needs to perform a series of equality queries to extract the data of a group of particles from each timestep based on their IDs and merge the data to define the temporal particle paths. The particle tracing is particularly expensive in the 3D case and requires here the majority of the time. The performance of the particle tracing was described earlier in Section 5.4.5. In a realistic use case scenario speed-ups of two orders of magnitude were achieved using FastBit compared to a sequential scan method. Besides the 3D histogram computation and the particle tracing also the minimum timepoint computation requires a considerable amount of time. This is due to the fact that the minimum timepoint computation needs to perform a hit count of how many particles satisfy the query $(px > 10^{10})$ at all timesteps. Using FastBit the algorithm is able to perform this hit count efficiently using only the bitmap index for *px* (without loading any raw data).

The actual analysis steps, i.e., segmentation, evaluation, merging, and path distance computation, are then very fast. This is mainly due to the overall structure of the pipeline allowing the independent analysis of each timestep. The performance of the bunch segmentation (and surface computation) is independent of the size of the dataset and depends only on the resolution of the underlying grid-based data structure (and the number of timesteps). As described in the previous section, the outstanding performance of the evaluation of the segmentation — in which the segmentation needs to identify the particles located within a set of histogram bins — is due to the bit vectors computed together with the 3D histograms which allow the segmentation to directly access the required data. The time spent for merging is also very short ($< 0.9s$) in all cases. The performance of the merging step largely depends on the number of bunch segmentations performed in the forward- and backward tracing. However, the expensive data preparation — consisting of the 3D histogram and surface computation — is at this point already completed for all timesteps so that the merging-function here only need to execute the much faster segmentation and evaluation step, explaining the short execution times of the merging in all cases.

5.7 Integrating Visualization and Analysis

In the case of LWFA research, simulations are usually performed on large supercomputers (here at NERSC) and the data produced is stored off-site at dedicated data storage facilities. The proposed automated analysis algorithms are, therefore, implemented as independent tools to enable the analysis to be executed where the data is stored. Separating the automated analysis tools from the visualization also has the advantage that the analysis algorithm can be optimized independently from the visualization resulting in lower development cost and likely also better runtime performance. However, for the purpose of the actual investigation of the analysis results dedicated links between the automated analysis and the visualization are needed.

In the context of the automated beam detection algorithm (see Section 5.5), currently the visualization capabilities of *R* are used for investigation of analysis results. In the case of the automatic beam path analysis (see Section 5.6), VisIt — the same system used for visual exploration of the raw data (see Section 5.4) — serves as main tool for visual inspection of analysis results. To integrate the automatic beam path analysis and the visualization of the raw data, the user can either: (i) visualize the output data of the analysis in the context of the raw data directly within VisIt, or (ii) create named selections based on the analysis results and apply the selections to the raw data for further analysis. Similar links could also be implemented between the automatic beam detection algorithm and VisIt but are not yet available.

To save analysis results the automatic beam path analysis creates for each detected particle bunch a beam path file, i.e., a VTK file [155] of all computed particle paths including information about the derived beam phases and path distances. In particular compared to the size of the original data, the size of the additional beam path files is usually small; approximately 1.5 to 60 MB per beam path file in the case of the examples presented in Section 5.6.7. The user can load and visualize the beam path files directly within VisIt. In VisIt the user can investigate analysis results using a variety of high-quality visualizations, such as, 2D and 3D particle path visualizations, scatter-plots, or 1D/2D and 3D histograms. VisIt can also display multiple visualizations of different datasets within the same viewer window. By visualizing analysis results in the context of the original data the user can effectively validate and investigate analysis results. Figure 5.35 and 5.36 as well as Figure 5.37 to 5.39 shown earlier were, e.g., created using this approach.

A second method for integrating the analysis with the visualization using VisIt is through the concept of named selections. As described earlier in Section 5.4.3.1, the concept of named selections allows for ID-based selection of particles within VisIt. Named selections, hence, allow the user to define persistent selections, i.e., selection that select the same particle subset at each timestep of a simulation. Based on the information derived in the beam path analysis a user can define a named selection — e.g., via thresholding in d_s and d_m — and apply it to the original raw data enabling detailed analysis of the so defined particle bunch. The user may chose to create such named selections in two different ways. First, the beam path analysis can automatically create a new named selection file for each detected particle bunch based on user-defined default thresholds in d_s and d_m. VisIt can then load the so defined named selections and make them accessible for the user. Second, a user may create named selections based on the information of the beam path files directly using VisIt. The first option has the advantage that it is faster and may, hence, be more appropriate in case that a larger number of datasets should be analyzed. The second option then has the advantage that the user can interactively define the particle beam of interest based on the path distance fields while receiving immediate feedback about the selection.

5.8 Results and Applications

Knowledge about the formation and temporal evolution of particles beams is paramount for the understanding, optimization, and development of next generation a laser wakefield particle accelerators. In order to gain a deeper understanding of the complex acceleration process, an effective analysis needs to address complex questions such as: i) which particles become accelerated; ii) how are particles accelerated, and iii) how was the beam of highly accelerated particles formed and how did it evolve [106]. Furthermore, in order to optimize simulation and experimental settings one needs to be able to: iv) investigate the quality of particle beams, and v) compare the behavior and quality of multiple particle beams. This section describes the use of the proposed methods to investigate these complex questions.

Section 5.8.1, describes how a user can identify and analysis particle beams in laser wakefield accelerator simulation using the proposed framework for visual exploration of the data (see Section 5.4). Following the use model described in Section 5.4.4, the user here first identifies the particles of interest via thresholding at a single timestep, then traces the particles over time to investigate the temporal evolution of the selected beam, and finally refines the selection based on information from different timesteps to investigate and compare particles subsets behaving differently over time.

Section 5.6.7 already showed that the automatic beam path analysis is able to effectively define particle beams. Section 5.8.2 now describes how a user can visualize and compare different particle bunches based on the results of the automatic beam path analysis visualized using VisIt. The described examples focus on the assessment the quality of particle beams and the analysis of their temporal evolution.

5.8.1 High-Performance Visual Analysis of Particle Beams

This section present a specific example illustrating the use of the proposed system for high-performance visual data analysis (see Section 5.4) to investigate 2D and 3D data produced by a laser wakefield particle accelerator simulation. The datasets used here contain more than $400,000$ and 90 million particles in the 2D and 3D datasets, respectively. The 2D data consists of 38 timesteps and has an overall size of about $1.3GB$, including the index structures. The 3D dataset consists of 30 timesteps and has an overall size of about $210GB$, including the index. Each 3D timestep has a size of about $7GB$, including the index, and about $5GB$ without the index. The example shown here first describes a detailed analysis of the 2D dataset and then extend the analysis to the larger 3D dataset.

Following the use model described in Section 5.4.4, the analysis is performed in multiple steps. First, to identify those particles that were accelerated, the user performs selection of particles at a late time point of the simulation by using a threshold for the value for x-momentum, px. By tracing the selected particles over time the user then analyzes the behavior of the beam during late timesteps (Section 5.8.1.1). Having defined the beam, the user analyzes the formation

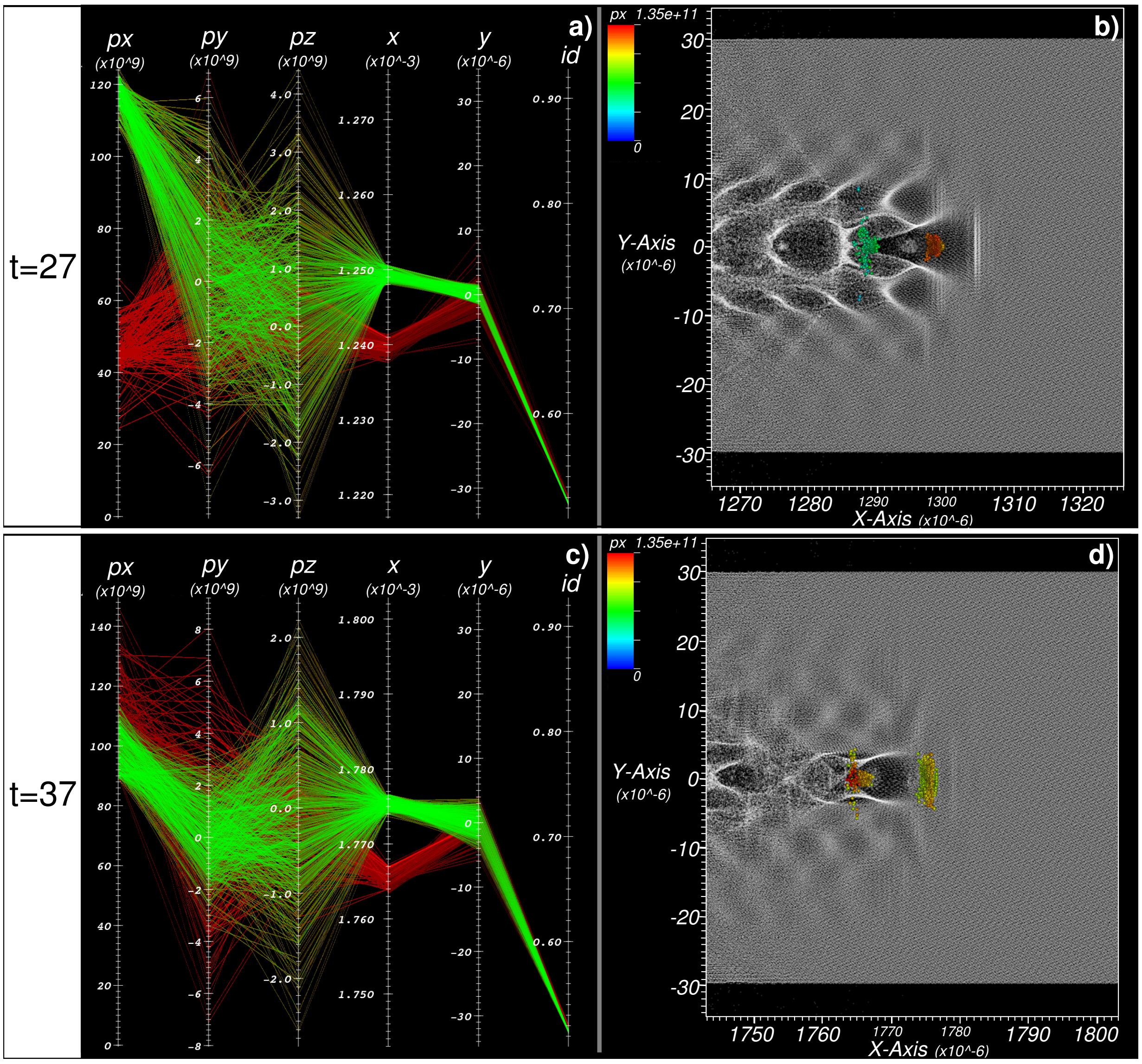


Figure 5.43: a) Parallel coordinates and b) pseudocolor plot of the beam at $t = 27$. Corresponding plots c,d) at $t = 37$. The context plot, shown in red, shows both beams selected by the user after applying a threshold of $px > 8.872 * 10^{10}$ at $t = 37$. The focus plot, shown in green, indicates the first beam that is following the laser pulse. The pseudocolor plots b) and d), show all particles in gray and the selected beams using spheres colored according to the particle's x-momentum, px. The focus beam is the rightmost bunch in these images. At timestep $t = 27$, the particles of the first beam (green in figure a) show much higher acceleration and a much lower energy spread (indicated via px) than the particles of the second beam. At later times, the lower momentum of the first beam indicates it has outrun the wave and moved into the decelerating phase, e.g., at timestep $t = 37$.

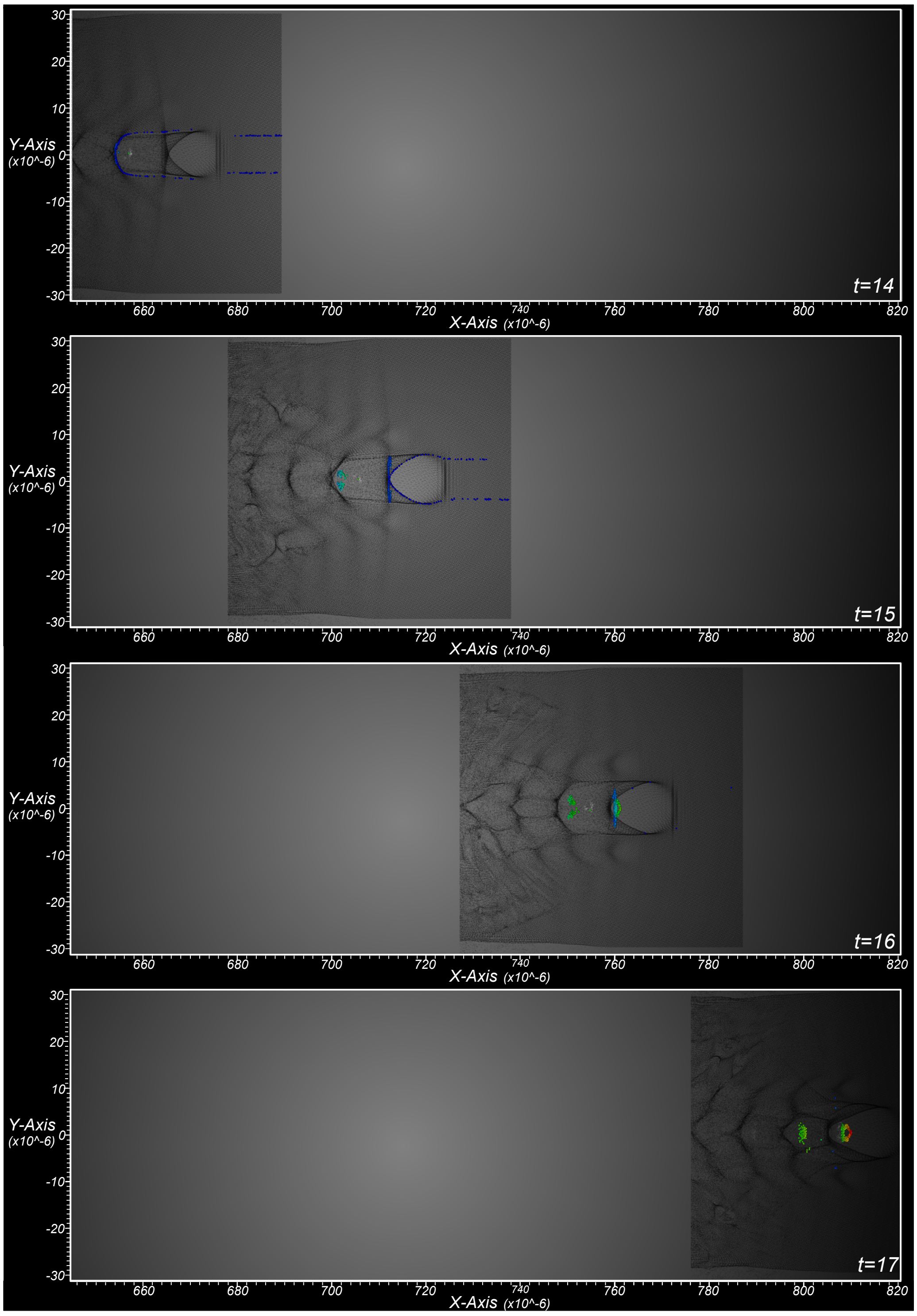


Figure 5.44: Particles of the beam at timestep 14 (top left) to 17 (bottom right). The color of selected particles indicates x-momentum, *px* (blue=low, red=high), while non-selected particles are shown in gray. Two main sets of particles entering the simulation at timestep $t = 14$ and additional sets of particles entering at $t = 15$ can readily be identified.

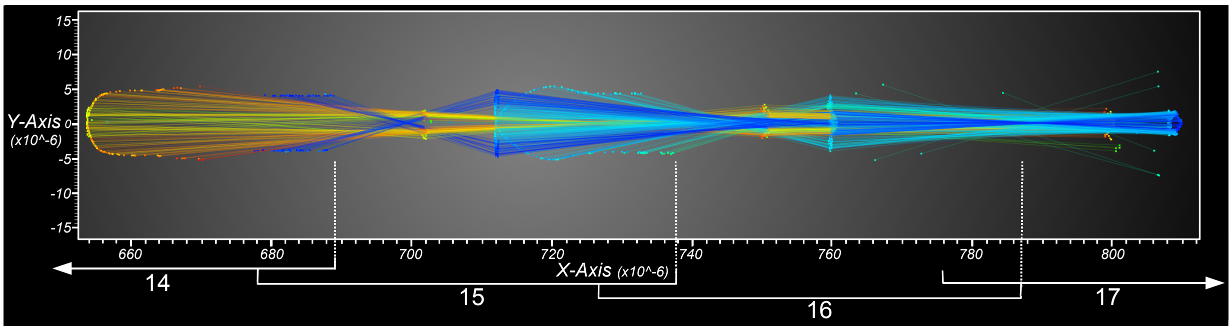


Figure 5.45: The traces of the beam particles during timesteps $t = 14$ to $t = 17$ as shown in Figure 5.44. The positions of the selected particles at the different times are also indicated by additional data points as well as annotations at the bottom of the figure indicating the begin and end of the timestep in x direction. Here, color is used to indicates particle ID.

and evolution of the beam by tracing particles further back to the time where they entered the simulation and became injected into the beam (Section 5.8.1.2). In this context, selection of particles, tracing of particles over time, as well as refinement of particle selections based on information from different timesteps are used as the main analysis techniques.

As this use case will illustrate, the ability to perform selection interactively and immediately validate selections directly in parallel coordinates as well as other types of plots, such as pseudocolor or scatter-plots, enables much more accurate selection than previously possible. Tracing of particles over time then enables researchers to better understand evolution of the beam. Using the proposed system based on the combination of VisIt, FastBit, and parallel coordinates, the user can rapidly identify subsets of particles of interest and analyze their temporal behavior. For "small" datasets one will usually perform all analysis on a regular workstation while for larger datasets VisIt can also be run in a distributed fashion so that all heavy computation is performed on a remote machine where the data is stored and only the viewer and the GUI executed on a local workstation.

5.8.1.1 Beam Selection

In order to identify the beam, i.e., find those particles that became accelerated, the user here first concentrates on the last timestep of the simulation (at $t = 37$). Using the parallel coordinates display, the user extracts the particles of interest by applying a threshold of $px > 8.872 * 10^{10}$. As is visible in the parallel coordinates plot, those particles constitute two characteristic clusters in x direction (Figure 5.43 c). Using a pseudocolor plot of the data, the physical structure of the beam can effectively be analyzed (see Figure 5.43 d).

Tracing particles back in time, then shows that the highest momentum and lowest momentum

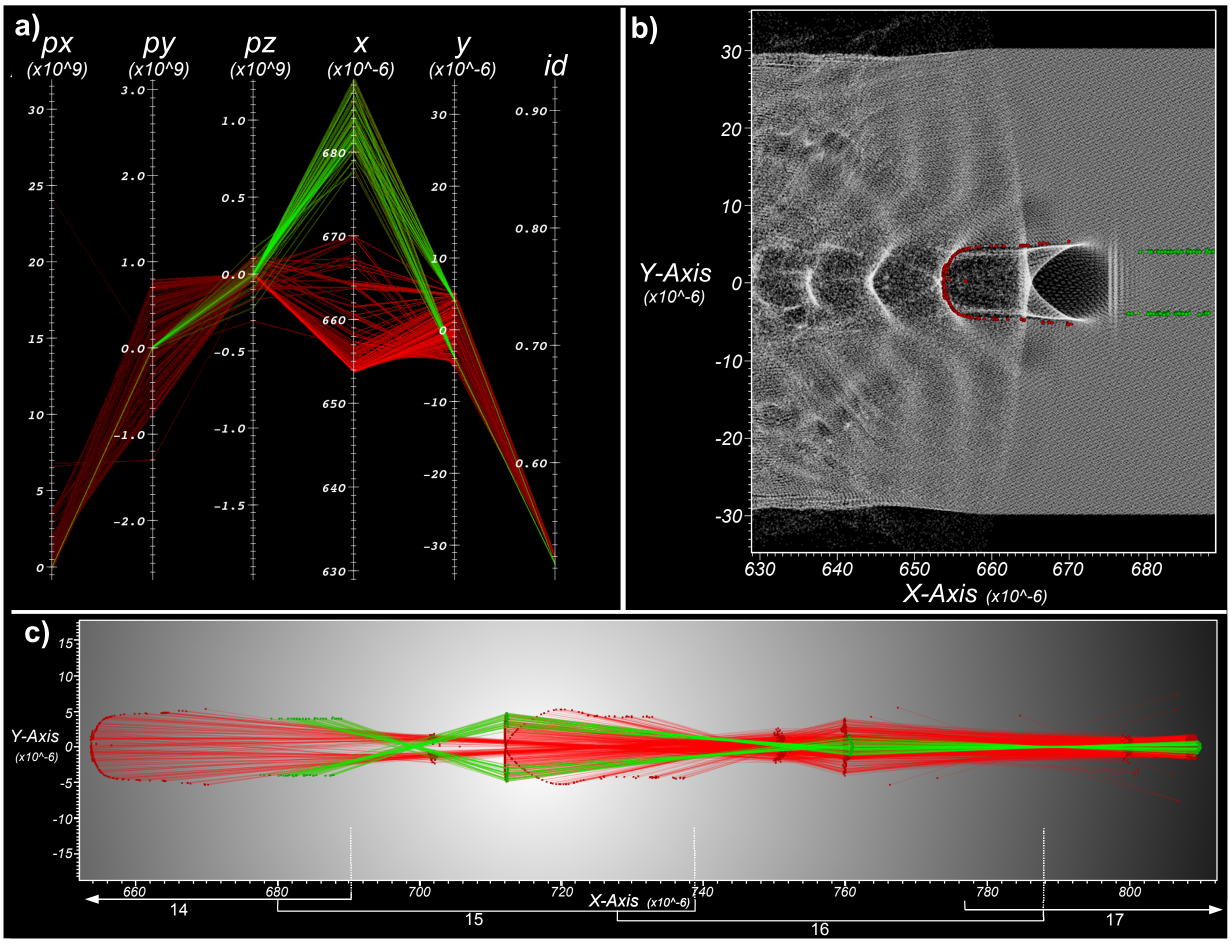


Figure 5.46: a) By applying an additional threshold in x at timestep $t = 14$, the user separates the two different sets of particles entering the simulation. b) The refinement result, shown in physical space, includes all non-selected particles (gray) to provide context. c) Particle traces of the complete beam and the refined selection. All plots show the complete beam in red and the refined selection in green. After entering the simulation, the selected particles (green) define first the outer part of the first beam at timestep $t = 15$. Later on at timesteps $t = 16$ and $t = 17$, these particles become highly focused and define the center of the first beam.

spread bunch observed in the simulation is formed at timestep $t = 27$ by the first bunch following the laser pulse (rightmost in these plots). In practice, the first beam following the laser pulse is typically the one of most interest to the accelerator scientists. The fact that the second beam shows higher or equal acceleration at the last timestep of the simulation is due to the fact that the first beam will outrun the wave later in time and therefore switches into a phase of deceleration while the second beam is still in an acceleration phase. In practice, when researchers want to select only the first beam, they usually perform selection of the particles using thresholding in px at an earlier time, e.g., $t = 27$, rather than the last timestep, here $t = 37$. By performing selection

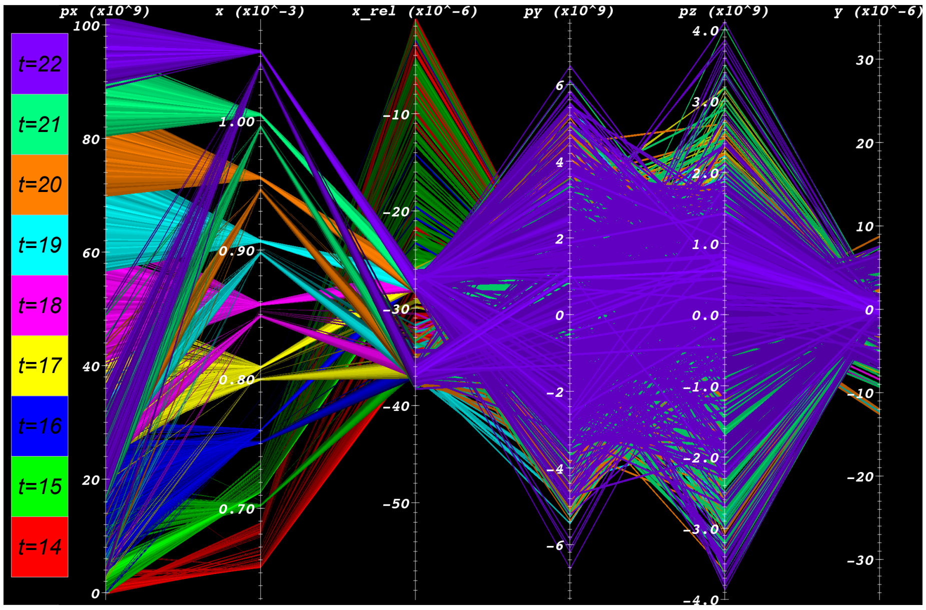


Figure 5.47: The beam at timesteps $t = 14$ to $t = 22$ in a temporal parallel coordinates plots. Here, color indicates each of the discrete timesteps. The plot shows clearly the two different beams in x and x_{rel}. While the second beam shows equal to higher values in px during early timesteps ($t = 14$ to $t = 17$), the first beam shows much higher acceleration at later times compared to the second beam.

at an earlier time, one avoids selecting particles in the second beam while being sure to select all particles in the first beam. In this specific use case the user is, however, interested in analyzing and comparing the evolution of these two beams, which is why selection was performed at the last timestep.

Figure 5.48(a and b) shows a similar example for the 3D particle dataset. At a much earlier time $t = 12$ ($x \approx 0.57 * 10^{-3}$ compared to $x \approx 1.3 * 10^{-3}$ in the 2D case) particles are trapped and accelerated, and the user selected particles in the first bunch via thresholding based on the momentum in x direction ($px > 4.586 * 10^{10}$) and x position ($x > 5.649 * 10^{-4}$) to exclude particles in the secondary waves from the selection. In order to get an overview of the main relevant data, the user removed the background from the data first by applying a threshold of $px > 2.0 * 10^{9}$. Figure 5.48b shows a volume rendering of the plasma density along with the selected particles revealing the physical location of the selected beam within the wake.

5.8.1.2 Beam Analysis

Having defined the beam in the 2D dataset, the user then analyzes the formation of the beam by tracing the selected particles back to the time when particles entered the simulation and were then injected into the beam. In Figure 5.44, the individual particles of the beam are shown at timesteps $t = 14$ to $t = 17$. Here, color indicates the particles' momentum in the x direction (px). Figure 5.45 shows the particle traces over time colored according to particle ID's. Different sets of injection are readily visible, and two sets of particles appear at $t = 14$. The left bunch will be injected to form a beam in the second wake period, which is visible at $t > 14$. A second group of particles is just entering the right side of the box (recall that the simulation box is sweeping from left to right with the laser, so that plasma particles enter it from its right side). This bunch continues to enter at $t = 15$, and the particles stream into the first (rightmost) wake period. These particles are accelerated and appear as a bunch in the first bucket for $t > 15$. At the following timesteps $t = 16$ and $t = 17$, further acceleration of the two particle beams can be seen while only a few additional particles are injected into the beam, and these are less focused.

Based on the information at timestep $t = 14$, the user refines the initial selection of the beam. By applying an additional threshold in x, the user selects those particles of the beam that are injected into the first wake period behind the laser pulse (see Figure 5.46a and 5.46b). By comparing the temporal traces of the selected particle subset (green) with the traces of the whole beam (red), one can readily identify important characteristics of the beam (see Figure 5.46c). After being injected, the selected particle subset (green) first defines the outer part of the first beam at timestep $t = 15$, while additional particles are injected into the center of the beam. Later on at timesteps $t = 16$ and $t = 17$, the selected particles become strongly focused and define the center of the first beam. By refining selections based on information at an earlier time, the user is able to identify characteristic substructures of the beam.

Temporal parallel coordinates then make possible the analysis of the general evolution of the beam in multiple dimensions (see Figure 5.47). Along the x axis, two separate beams can be seen at all timesteps ($t = 14$ to $t = 22$) with a quite stable relative position in x (x_{rel}). At early timesteps, both beams show similar acceleration in px while later on, at timestep $t = 18$ to $t = 22$ (magenta to lilac), the particles of the first beam show significantly higher acceleration with a relatively low energy spread. Particles with a relatively high relative position in x direction are found mainly at timesteps $t = 14$ and $t = 15$ due to the fact that new particles enter the simulation window mainly during these times.

Figure 5.48c shows an example for the 3D dataset. Here, the traces of the particles selected earlier in Figure 5.48a are shown. The user selected the particles at timestep $t = 12$ then traced them back to timestep $t = 9$ where most of the selected particles enter the simulation window and forward in time to timestep $t = 14$. As one can see in the plot, the selected particles are constantly accelerated over time.

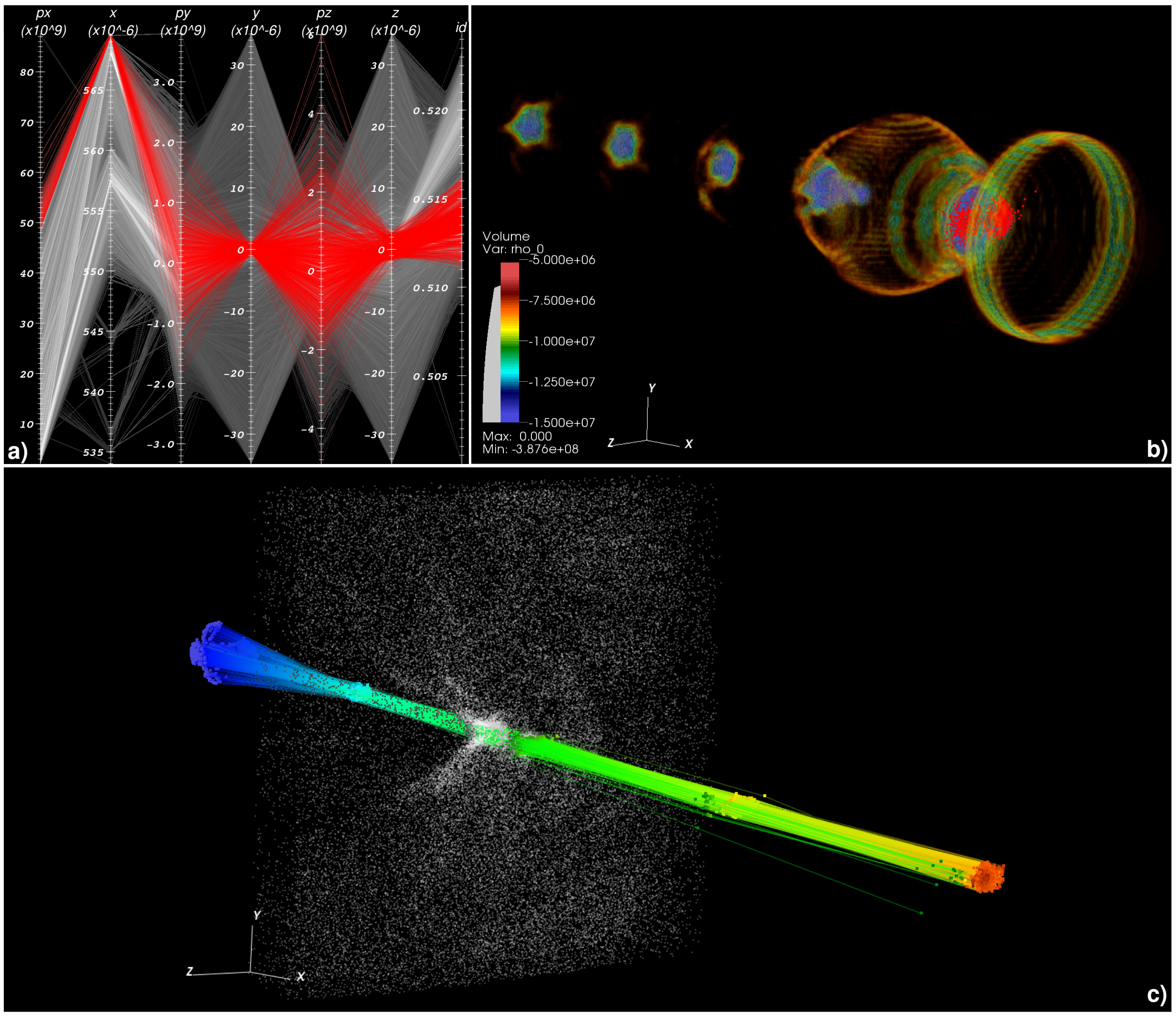


Figure 5.48: a) Parallel coordinates of timestep $t = 12$ of the 3D dataset. Context view (gray) shows particles selected with $px > 2 * 10^9$. The focus view (red) consists of particles selected with $px > 4.856 * 10^{10}$ && $x > 5.649 * 10^{-4}$, which indicates particles forming a compact beam in the first wake period following the laser pulse. b) Volume rendering of the plasma density and the selected focus particles (red). c) Traces of the beam. The user selected particles at timestep $t = 12$, then traced the particles back in time to timestep $t = 9$ when most of the selected particles entered the simulation window. The user then also traced the particles forward in time to timestep $t = 14$. In this image, color is used to indicate px (blue=low, red=high). In addition to the traces and the position of the particles, the image also shows the context particles at timestep $t = 12$ in gray to illustrate where the original selection was performed. The figure shows that the selected particles are constantly accelerated over time (increase in px).

5.8.2 Beam Comparison and Visualization

Section 5.6 showed that the automatic beam path analysis is able to detect the main bunches of interest in the example dataset. Having detected the particle bunches of interest one main question is which bunch has the highest quality. This section demonstrates how one can investigate and compare the quality of particle bunches based on the results from the automatic beam path analysis (see Section 5.6). The quality of a bunch is characterized by several factors. A high-quality beam should: (i) have low energy spread (indicated by its compactness in px); (ii) be condensed in physical space; (iii) reach high energy levels (indicated by high px); and (iv) be focused (indicated by low transverse momentum py and in 3D also pz).

Using the path distance functions d_s and d_m one can effectively compare the compactness of different bunches of the same simulation simply by comparing their respective histograms. Figure 5.49a, shows, e.g., the histograms of the distance in momentum space d_m for the two bunches detected in the example dataset. For the first bunch (red) many more particles with low values of d_m exist than for the second bunch, i.e., the first bunch will be more compact in momentum space and therefore shows a lower energy spread. Similarly, the first bunch can also be seen to be more condensed in physical space than the second bunch by comparing the histograms of the distance in physical space d_s (see Figure 5.49b). These findings indicate that the first bunch is highly condensed in physical as well as momentum space and is therefore of higher quality than the second bunch.

By comparing the temporal paths of all particles that form the two beams one can assess which energy levels the two bunches achieve and how focused they are. To define the two particle

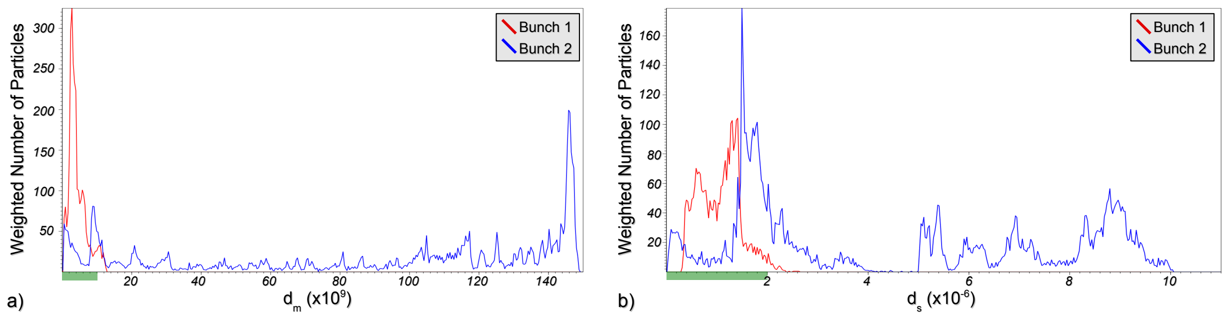


Figure 5.49: a) Histogram of the distance in momentum space d_m of all candidate particles detected for each of the two bunches. b) Histogram of the distance in physical space d_s of all candidate particles detected for each of the two bunches. The main region of interest below the default thresholds in d_m and d_s respectively is indicated in green in both figures. For the first bunch many more particles with low d_m and d_s values exist indicating that this beam is more condensed in both physical and momentum space and therefore of higher quality.

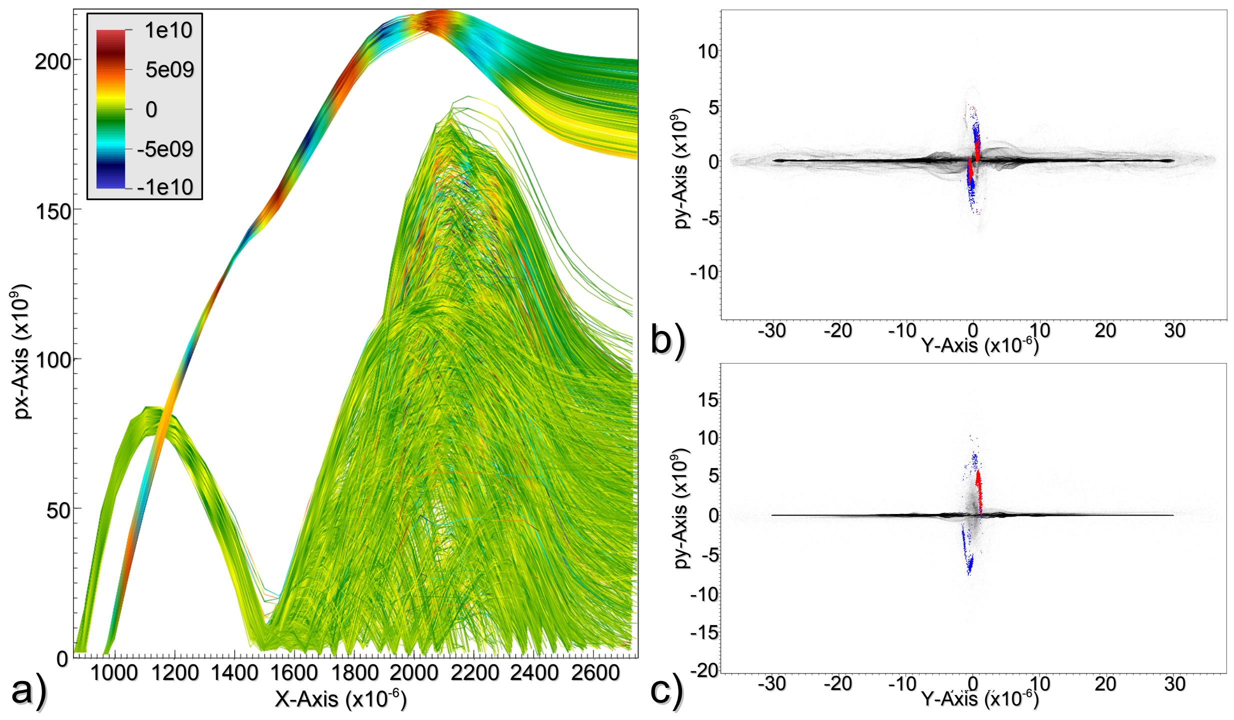


Figure 5.50: a) Paths of both bunches in x/px space after applying the thresholds $(d_s < 10^{-6})\&\&(d_m < 10^{10})$. Paths are colored according to py using the indicated color mapping for both bunches. Although the second bunch achieves overall higher levels of acceleration the first bunch shows much less variation in py while it exists. b) Density plot of y/py space at $t = 24$ (gray) and all particles of the first bunch with $(d_s < 10^{-6})\&\&(d_m < 10^{10})$ shown in red and with $(d_s < 2 * 10^{-6})\&\&(d_m < 10^{10})$ shown in blue. c) Density plot of y/py space at $t = 43$ (gray) with particles of the second bunch shown using the same coloring scheme as in *b*. The timesteps $t = 24$ and $t = 43$ were selected because these are the timesteps at which the according bunch has its peak momentum in px. When comparing figure *b* and *c* one can see that the first bunch is much more condensed in y/py space than the second bunch.

beams corresponding thresholds in d_m and d_s are applied first, using the same thresholds for each of the two bunches. Figure 5.50a shows the temporal paths of the selected particles in x/px space colored according to the transverse momentum py. The second bunch achieves higher levels of px than the first bunch but also shows much more variation in py. This indicates that while the second bunch reaches much higher energy levels than the first bunch it also shows more spread in py. The fact that the second bunch is less condensed than the first bunch is even more apparent when comparing their structure in y/py space at the time when they reach their peak energy. Figure 5.50b shows the respective plot for the first bunch at timestep $t = 24$ and Figure 5.50c for the second bunch at timestep $t = 43$. When comparing the two plots one can clearly see that the second bunch is much less condensed in y/py space at its peak energy than the first bunch.

Overall this analysis indicates that the first bunch: i) has a low energy spread, ii) is condensed in physical space, and iii) is highly focused. Even though the first bunch has a lower peak energy these findings indicate that it is in certain parameters of higher quality than the second bunch. The proposed beam path analysis method not only enables the separation of different bunches but also the analysis of their quality and processes contributing to the bunch-quality.

Investigation of the paths of the beam particles also enables analysis of the temporal evolution of the two particle beams. From the beam phases computed in the reference path analysis (see Figure 5.33) one already knows that the first bunch is formed earlier than the second bunch. It then accelerates over a short period of time and then outruns the wave relatively quickly and switches into the deceleration phase. The second bunch on the other hand is formed later and is accelerated over a longer period of time. When comparing the traces of the two beams shown in Figure 5.50a one can also see that the two bunches show a very different acceleration/deceleration behavior. While the paths of the first bunch show a characteristic horseshoe-like shape the second bunch shows a less smooth acceleration behavior and then decelerates more gradually than the first bunch.

Using 3D particle path visualizations one can effectively investigate the temporal evolution of particle beams in up to four data dimensions at once. Figure 5.51 shows, e.g., the particle paths of the first bunch in $x/y/py$ space colored according to px. Figure 5.52 shows an example of relative particle paths for a bunch detected in dataset D (see Table 5.2). Visualizations of relative instead of absolute particle paths allows investigation of the motion of particles within the simulation window. Using VisIt a user can then also effectively analyze the temporal evolution of particle beams via animations of the complete time series.

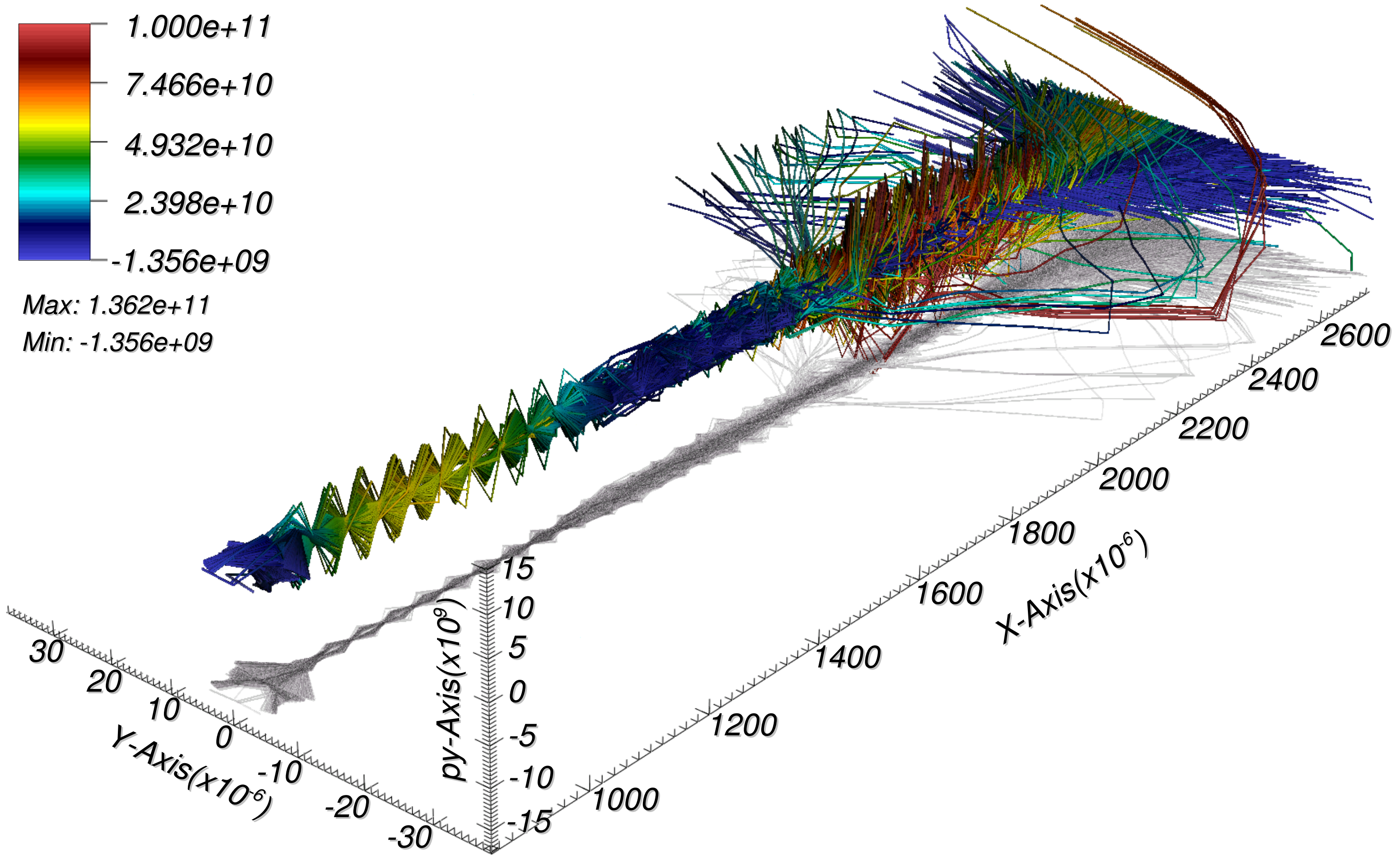


Figure 5.51: Paths of the first bunch in $x/y/py$ space after applying the thresholds $(d_s < 2 * 10^{-6})\&\&(d_m < 10^{10})$. Paths are colored according to px using the indicated color mapping. The bunch reaches its peak energy at $x \approx 1200 * 10^{-6}m$, afterwards the bunch decelerates. At $x > 1500 * 10^{-6}m$ the bunch has lost its coherency, while some particles leave the simulation window and are no longer traceable, others undergo a secondary phase of acceleration. While it exists the beam particles show a very regular motion in y/py. Particle path visualizations in 3D enable investigation of the evolution of particle beams in up to four dimensions.

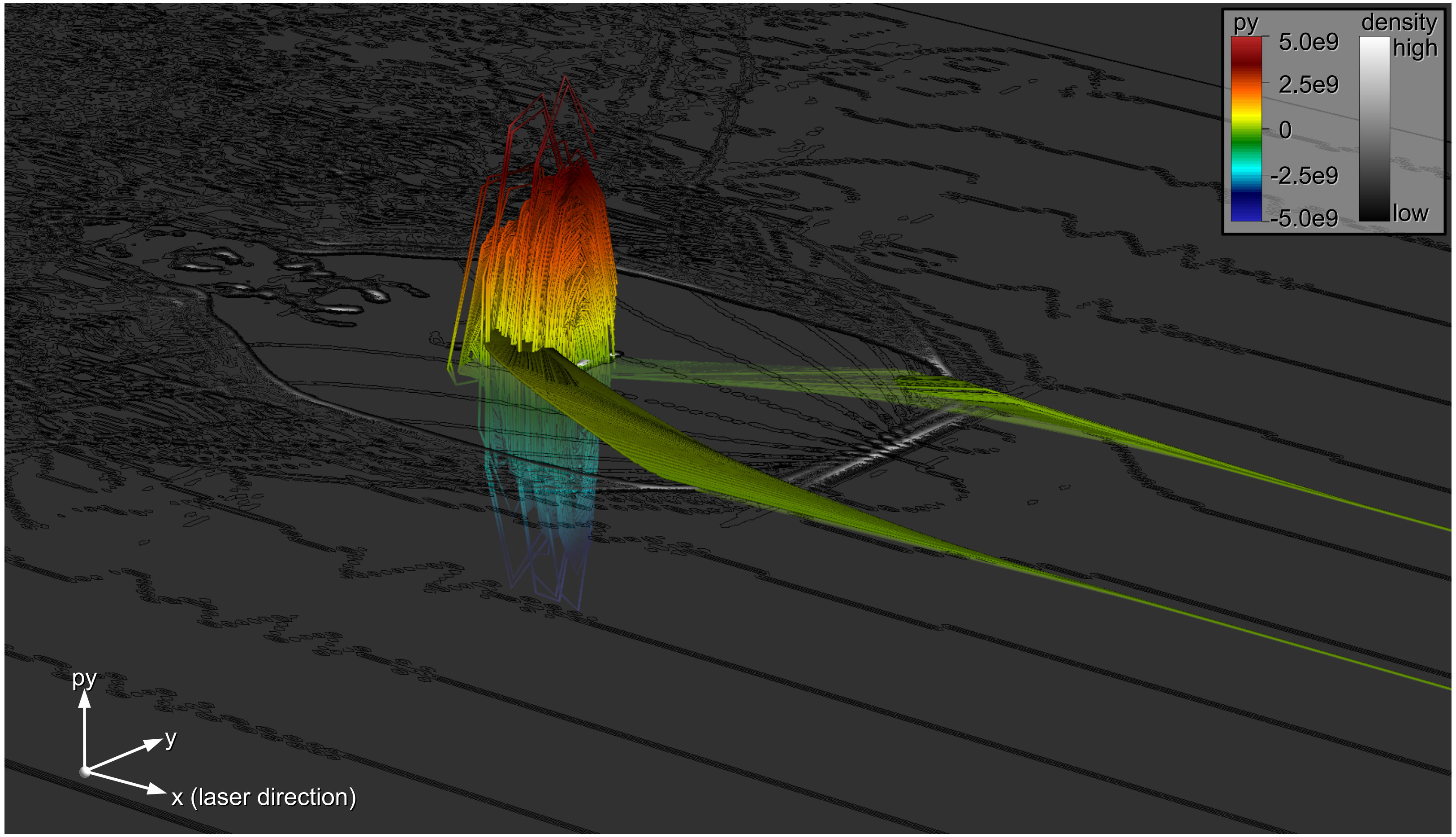


Figure 5.52: Relative particle paths of the first bunch detected in a high-resolution 2D dataset (see also Figure 5.38(D1)) shown in $x/y/py$ space colored according to py. Instead of the absolute x location the visualization here uses the relative x location of particles within the simulation window to illustrate the motion of particles within the wake of the laser. Traces are shown up to timestep $t = 95$ when the bunch reaches its peak momentum in px. On the x/y plane iso-contours of the particle density at timestep $t = 95$ are shown. One can see particles being injected from the sides and oscillating while being accelerated.

Chapter 6

Conclusions

The overall objective of this research effort was the development of new capabilities to accelerate scientific knowledge discovery from complex, large, multivariate scientific data. An analysis framework based on the integration of data analysis and visualization has been proposed to enable efficient analysis of scientific data. Using two distinct example applications, this work has demonstrated the effectiveness of the proposed approach to address relevant problems in developmental biology (Section 6.1) and high-energy physics (Section 6.2)

6.1 Visualization and Analysis of 3D Gene Expression Data

The proposed framework for knowledge discovery from 3D gene expression data helps biologists who aim to discover potentially new, experimentally verifiable biological interactions, by providing them with the ability to define, analyze, and iteratively refine clusters in multiple, linked views. For computational biologists, objective methods for classifying quantitative data points in spatial datasets have been presented.

This research has shown how data clustering and visualization can be integrated into one framework and how the developed system can be used effectively to explore and analyze 3D spatial expression data. The analysis system MATLAB was integrated with the visualization providing biologists seemless access to advanced data analysis methods and enabling bioinformatics researchers to integrate their analysis directly with the visualization.

A system of linked multiple views has been presented that is used for data exploration and for steering the analysis process, helping bridge spatial patterns of expression with abstract views of quantitative expression information.

Data clustering then provides means for automatically defining cell selections, depicting characteristic data features and in this way improving the visualization. Dedicated post-processing of clustering results based on visualization and user knowledge has been demonstrated to improve the analysis. Proper definition of the number of clusters k is essential to ensure accurate grouping of cells into clusters. This research effort has shown how the combination of $\varepsilon_{\text{scatter}}$ as measure to describe the relative physical scattering of clustering results and ε_{exp} can be used to suggest a good initial value for k as well as a series of possibly good alternative values for k. In combination with visual validation of clustering results this approach has been shown to be effective for determining appropriate values for k.

Analysis of 3D spatial gene expression data is a challenging task requiring unique strategies not encountered in studies of 1D non-spatial data, such as microarray expression data. Using the proposed integrated data visualization and analysis approach, this research has shown how the pattern of a gene and its temporal variation can be defined and analyzed. Furthermore, this research has demonstrated how suspected relationships between genes can be analyzed to address the question of how the pattern of a gene is created by the action of multiple regulators.

Along with the first release of the BDTNP 3D gene expression database, the BDTNP has also made a version of PointCloudXplore (PCX) freely available to the public [91]. Data clustering and 3D parallel coordinates are today used regularly by BDTNP members and will soon also be included in the public version of PCX.

One focus of potential future research could be the development of additional analysis techniques that effectively integrate spatial and gene expression information. Adaptation of spatial clustering methods, such as the dual clustering approach proposed by Cheng-Ru et al. [156], is only one promising approach. Alternatively, one could perform clustering based on gene expression information only, split the resulting clusters into spatially distinct sub-clusters, and then perform a re-clustering based on the centers of the detected sub-clusters. PCX currently uses hierarchical clustering only for partitioning of the data. By traversing the data hierarchy created in a hierarchical clustering, exploration of the data at multiple levels of detail becomes possible. In addition to clustering of cells, clustering of genes as well as biclustering promises to provide further insights into the data. Integration of these and other data analysis techniques, such as expression boundary detection [94] or regression analysis to predict regulatory interaction [161], into PCX should further increase its value for practical use and impact.

6.2 Visualization and Analysis of Laser Wakefield Particle Acceleration Data

Managing and gaining insight from vast amounts of data is widely accepted as one of the primary bottlenecks in science today. The research presented in this thesis is directed at enabling rapid knowledge discovery from large, complex, multivariate, and time-varying LWFA simulation datasets and using modern HPC platforms.

To achieve this objective, a novel approach for quickly creating histogram-based parallel coordinates displays has been presented. This research leverages this form of visual information display as the basis for forming complex multivariate range queries. This form of visual information display has been combined with state-of-the-art index/query technology to quickly compute conditional histograms as well as to efficiently extract subsets of data that meet certain conditions. A performance study was conducted of each of the fundamental algorithms that form a complete implementation in a production-quality, visual data analysis application with support for parallel computation. The study presented has shown the efficacy of the algorithms for computing histograms and for performing particle subset selection. These algorithms have been shown to have favorable scalability characteristics over a set of processor pool sizes ranging from 1 up to 100 on a modern HPC platform, a Cray XT4. From the scientist's point of view, it is now possible to perform a type of visual data analysis in a few seconds with this new framework as compared to several hours required using legacy tools. These results are significant because they show the successful synergy that can result when combining visual data analysis with scientific data management technologies to solve challenging problems in scientific knowledge discovery.

Two novel methods for automatic analysis of particle beams in datasets generated by LWFA simulations have been presented. Automatic beam detection, a novel approach based on bunch lifetime analysis and fuzzy clustering, has been demonstrated to be effective for detecting the highest-energy particle bunch in datasets produced by LWFA simulation. This thesis has also introduced automatic beam path analysis, a novel approach for automatic detection and analysis of multiple particle beams based on the complete temporal path of the particles that form them. A detailed study of the performance of the automatic beam path analysis was conducted demonstrating that the method is able to efficiently detect particle bunches of interest even in extremely large 3D simulation datasets.

Via integration of the introduced automatic analysis methods with visualization, the approaches developed in this research enable efficient analysis of extremely large 3D particle datasets. A detailed case study showed how these technologies can be used in concert to explore large, time-varying LWFA simulation datasets addressing relevant issues, such as (i) detection and definition of particle beams, (ii) analysis of the formation and acceleration of particle beams, (iii) investigation of the quality of particle beams, and (iv) comparison of the behavior and quality of multiple particle beams.

Concerning possible directions for future research, one might want to include the exploration of different avenues for parallelizing the most expensive parts of the visual data analysis processing pipeline. As visual data analysis is typically an interactive process, minimizing the response time is crucial to maximize effectiveness. Similarly, parallelizing the automated data analysis procedures would likely make possible much more efficient analysis of large collections of simulation datasets. The research presented in this thesis was mainly focused on the analysis of particle data while the field data is mainly used to provide context in the visualization. Development of new methods to provide closer links between the particle and the field data holds the potential to further increase the value and impact of the research presented in this thesis.

Appendix A

PointCloudXplore: Interface to MATLAB

This appendix provides an overview of the PCX/MATLAB interface. Section A.1 and A.2 first describe how to integrate a function defined in MATLAB into PCX. Section A.3 then gives an overview of the used named conventions a developer has to consider when defining a MATLAB function for PCX. A set of simple examples illustrating the functionality and different applications of the interface are presented in A.4.

A.1 Defining a MATLAB Function for PointCloudXplore

Defining a new MATLAB function that can be accessed directly from PCX involves three simple steps. First, one needs to create a *M* file (file extension *.m*) specifying the MATLAB function. M files can be created either directly in MATLAB or using a simple text editor. In order to be accessible from PCX the definition of the function must follow the name conventions described in Section A.3. Second, to make the new function accessible for PCX one needs to define a *PCXM* file describing how PCX should call and represent the function (see Section A.2). Third, to inform PCX that a new MATLAB function is available one needs to either: i) copy the *PCXM* file into PCX's default *FUNCTIONS* folder, or ii) set the function path using the according *"Set Function Path"* option available in the MATLAB menu located in the main menu bar of PCX.

At start-up PCX parses all PCXM files located at the last specified function path. Based on the information defined in the PCXM files PCX then automatically creates for each function an according entry in the MATLAB menu bar. The GUI used to specify the input parameters and data for a function is created on the fly at runtime when the user calls the function. Since all components are created at runtime one can add or modify MATLAB functions while PCX is running without the need to restart the program. Using the *"Refresh"* or *"Set Function Path"* option available in the main menu bar of PCX one can inform PCX that a MATLAB function has been changed or added. PCX then parses all PCXM files located at the current function path and updates its MATLAB menu accordingly.

A.2 PCXM File Specification

A PCXM file specifies: i) How PCX should represent the MATLAB function in the GUI; ii) How PCX can access the function; iii) Which input parameters the function expects, i.e., which data does PCX need to export to MATLAB upon calling the function; and iv) Which output the function creates, i.e., which data does PCX need to import from MATLAB after the function has complete execution. PCX expects the file extension *.pcxm* for PCXM files. PCXM files are text files that can be created using any text editor. In a PCXM files the symbol ":" is used as separator, i.e., after each ":" a new entry is expected to appear. Table A.1 describes the structure of a PCXM file.

Table A.1: PCXM file specification.

Specifier	**Description**
TYPE:	Type of the MATLAB function., e.g., "Plot" or "Filter". PCX presents all function that have the same type together in a submenu of the MATLAB menu.
NAME:	Name of the function as it should appear in the main menu.
DIR:	Path to the directory where the M file is located specifying the function to be called. Can be empty in case the M file is located in the same folder as the PCXM file.
CALL:	Defines which command PCX should issue to MATLAB in order to call the function. Based on this string PCX also identify which data it needs to transfer to MATLAB before calling the function and import after completion of the function. This call therefore needs to follow the name conventions for the input/output parameters as specified below (see Section A.3).

IN_GENES:	Specifies the maximum number of genes the MATLAB function takes as input. Leave empty in case that the number of genes the function takes as input is not limited. This parameter is ignored in case that genes are not an input parameter of the function.
IN_BRUSHES:	Specify the maximum number of Cell Selectors the MATLAB function takes as input. Leave empty in case that the number of Cell Selectors the function takes as input is not limited. This parameter is in case that Cells Selectors are not an input parameter of the function, i.e., *input_brushes* is not specified (see Section A.3).
IN_DOUBLE:	Specifies a list of additional double input parameters required by the MATLAB function. For each parameter specify the name and which text should appear in the GUI separated by a ":". Leave empty in case that no additional double input parameters are required. NOTE: Send to MATLAB as *mxDOUBLE_CLASS*.
IN_INT:	Specifies a list of additional integer input parameters required by the MATLAB unction. For each parameter specify the name and which text should appear in the GUI separated by a ":". Leave empty in case that no additional double input parameters are required. NOTE: Send to MATLAB as *mxINT32_CLASS*.
IN_BOOL:	Specifies a list of additional Boolean input parameters required by the MATLAB function. For each parameter specify the name and which text should appear in the GUI separated by a ":". Leave empty in case that no additional double input parameters are required. NOTE: Send to MATLAB as *mxINT32_CLASS*.

A.3 Name Conventions

In order to enable PCX to call a MATLAB function properly the PCXM header file needs to follow the here described name conventions. The function call to be issued by PCX (specified in CALL: in the PCXM file described above) should look as follows:

$$[output\_genes, output\_genenames, output\_brushes, output\_brushnames] = \\ functionname \\ (input\_genes, input\_genenames, input\_brushes, input\_brushnames, \\ input\_cellpos, input\_unrolledpos, input\_cellneighbors, input\_filename, \text{``}otherinputparameters''\text{)}$$

The output parameters of the function to be imported by PCX are shown in blue and the input parameters of the function to be send from PCX to MATLAB are shown in green. All input and output parameters are optional. Based on whether the function call contains the key words described in Table A.2 PCX identifies which actions it needs to take in order to call the function properly. In general any combination of input and output parameters is valid. However the following parameters are treated as dependend: (i) *input_genenames* is only send if *input_genes* is specified; (ii) *input_brushnames* is only send if *input_brushes* is specified; (iii) *output_genenames* is ignored in case that *output_genes* is not specified; (iv) *output_brushnames* is ignored in case that *output_brushes* is not specified.

Table A.2:Name conventions of the PCX-MATLAB interface.

Name	**Description**
input_genes	Matrix (2D-array) with the expression information of the genes. **Send:** 2D array of type *mxDOUBLE_CLASS* of size $m \times n$ with m being the number of input genes and n being the number of cells. **Access in MATLAB:** $genei = input\_genes(i,:)$ returns the expression level of the i'th gene send to MATLAB in all cells.
input_genenames	String array with the names of the input genes. **Send:** 2D array of type *mxCHAR_CLASS* **Access in MATLAB:** $genei = input\_genenames(i,:)$ returns the name of the i'th gene send to MATLAB. **Note:** This parameter is ignored in case that no genes are send (input_genes not specified).

Table A.2:Name conventions of the PCX-MATLAB interface.

Name	**Description**
input_brushes	Matrix (2D-array) with the selection information of the cell selections used as input. A cell selector may define a smooth selection. Selection values should always be in the range of $[0,1]$ where 0 indicates that a cell is not selected and 1 that a cell is selected. **Send:** 2D array of type $mxDOUBLE\_CLASS$ of size $m \times n$ with m being the number of cell selections sent and n being the number of cells of the embryo. **Access in MATLAB:** $brushi = input\_brushes(i,:)$ returns the selection filter array of the i'th cell selector send to MATLAB defining which cells are selected.
input_brushenames	String array with the names of the input Cell-Selectors. **Send:** 2D array of type $mxCHAR\_CLASS$ **Access in MATLAB:** $brushi = input\_brushames(i,:)$ returns the name of the i'th Cell Selector send to MATLAB **Note:** This parameter is ignored in case that no Cell Selectors are send (i.e., *input_brushes* is not specified).
input_cellpos	Send x,y,z cell positions to MATLAB. **Send:** 2D array of type $mxDOUBLE\_CLASS$ of size $3 \times n$ with n being the number of cells of the embryo. **Access in MATLAB:** The following MATLAB calls return the x,y,z position of all cells of the embryo respectively: $x = input\_cellpos(1,:)$; $y = input\_cellpos(2,:)$; $z = input\_cellpos(3,:)$
input_unrolledpos	Send the AP/DV positions derived from the Unrolled View. The data is send as 2*x*n array with *n* being the number of cells. **Send:** 2D array of type $mxDOUBLE\_CLASS$ of size $2 \times n$ with n being the number of cells of the embryo. **Access in MATLAB:** The following MATLAB calls return the projected AP, DV position of all cells of the embryo respectively: $ap = input\_cellpos(1,:)$; $dv = input\_cellpos(2,:)$

Table A.2:Name conventions of the PCX-MATLAB interface.

Name	**Description**
input_cellneighbors	Send array with the neighborhood information of all cells as defined by the voronoi diagram used in the CellView of PCX. **Send:** 2D array of type *mxDOUBLE_CLASS* of size $n \times m$ with n being the number of cells of the embryo and m being the maximum number of neighbors of a cell. For each cell that has less than the maximum number of neighbors entries of -1 are added to fill the rows of the matrix. **Access in MATLAB:** $neighi = input\_cellneighbors(i,:)$ returns all neighbors of cell the with index i. First all valid neighbors are listed. An entry of -1 indicates that the cell does not have any more neighbors. **Note:** In order to be able to send a matrix with the same number of entries in each row -1 entries are added as neighbors to cells which do not have the maximum number of neighbors. All entries larger than -1 indicate valid neighbors.
input_filename	Send the name (and path) of the current PointCloud file to MATLAB **Send:** 2D array of type *mxCHAR_CLASS* of size $1 \times m$ with m being the length of the filename string. **Access in MATLAB:** $filename = input\_filename$ **Note:** In case that PCX is started from the shell with a path as input option then the given input path is returned which may not be an absolute but a relative path. If the PointCloud was opened in the GUI then the full path is always send to MATLAB.
output_genes	Matrix (2D-array) with the newly created gene patterns to be imported by PCX. **Expected format:** 2D array of type *mxDOUBLE_CLASS* of size $m \times n$ with m being the number of genes returned and n being the number of cells of the embryo.
output_genenames	List of strings with the names for the newly created gene patterns. **Expected format:** 2D array of type *mxCHAR_CLASS* with m entries with m being the number of *output_genes* returned.

Table A.2:Name conventions of the PCX-MATLAB interface.

Name	**Description**
	Note: This parameter is ignored in case that *output_genes* is not an output parameter of the MATLAB function. In case that *output_genes* is specified but *output_genenames* is not, then PCX automatically assigns a default name to the imported genes.
output_brushes	Matrix (2D-array) with the newly created Cell Selectors to be imported by PCX. **Expected format:** 2D array of type *mxDOUBLE_CLASS* of size $m \times n$ with m being the number of Cell Selectors returned and n being the number of cells of the embryo.
output_brushnames	List of strings with the names for the newly created Cell-Selectors. **Expected format:** 2D array of type *mxCHAR_CLASS* with m entries, with m being the number of Cell Selectors returned. **Note:** This parameter is ignored in case that *output_brushes* is not an output parameter of the MATLAB function. In case that *output_brushes* is specified but *output_brushnames* is not then PCX automatically assigns a default name to the imported Cell Selectors.

A.4 Examples

This sections describes a set of simple examples illustrating the use of PCX/MATLAB interface and some of its possible applications. Examples A.1 - A.6 show: i) a screenshot of PCX with the according result of the script, ii) the MATLAB code included in the according M file, and iii) the description included in the PCXM file. In most cases the screenshot also shows the GUI automatically created by PCX to allow the user to define the input for the MATLAB functions.

A.4.1 Plotting

MATLAB provides a rich set of visualizations which one may want to use to investigate the data. Examples A.1 - A.4 illustrate some simple gene expression plots (including cell locations) created using MATLAB.

A.4.2 Filtering

Gene expression filters derive one or multiple pseudo-expression patterns from a set of input gene expression patterns. In order to export the filtered values to PCX one needs to save the data in the *output_genes array*. Here (i,:) corresponds to all values for the newly created gene with index i. As illustrated in Sections 4.7.5 and 4.7.6, gene expression filters have a wide range of applications, such as, modeling of genetic regulatory networks, analysis of the temporal change of a pattern, comparison of expression patterns, or cluster validation.

A.4.3 Selection

MATLAB provides a rich set of data analysis tools (e.g. clustering) which one may want to use to detect specific features in the data. PCX can import these selection and save them as cell selectors. Cell selectors created via MATLAB can be analyzed using any view available in PCX. One can also use these selections in later MATLAB based analysis.

In order to create a selection in MATLAB one needs to create an array and define for each cell a value between $[0,1]$ and save the values in the *output_brushes* array. Here (i,:) corresponds to all values for the newly created cell selector with index i. In general the selection values should be either 0 (not selected) or 1 (selected). PCX in general allows any value between $[0,1]$ as filter value for cell selector. It is, however, not guaranteed that every view supports smooth selections properly, i.e., cell selectors with selection filer values other than 0, 1 properly. Some views may display all cells with a filter value larger than some threshold as selected rather than a smooth selection. Example A.6 defines such a smooth selection by copying the expression values of the input genes to the *output_brushes* array.

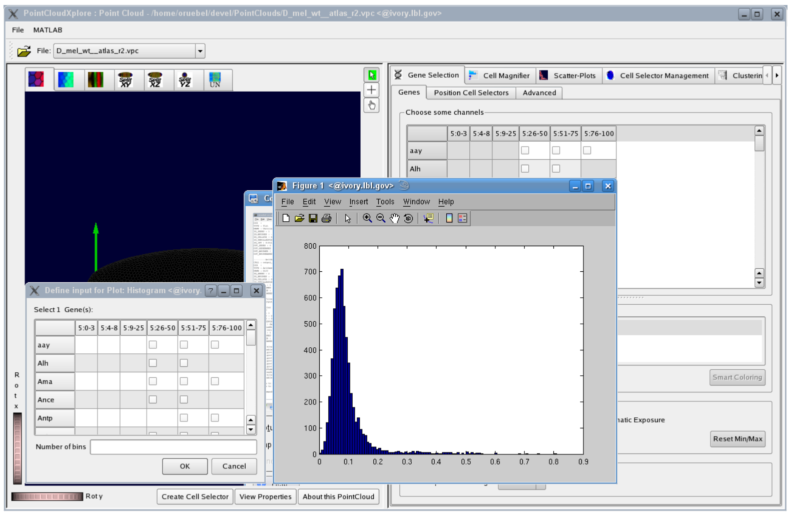


Figure A.1: **Gene expression histogram in MATLAB:** This example function creates a histogram of the gene expression values of a user defined gene using a user specified number of bins.

Histogram.m:

```
function Histogram(input_genes , nbins)
n = cast(nbins , 'double');
hist(input_genes,n);
```

Histogram.pcxm

```
TYPE:Plot
NAME:Histogram
DIR:
CALL:Histogram(input_genes, nbins)
IN_GENES:1
IN_BRUSHES:0
IN_DOUBLE:
IN_INT:nbins: Number of bins
IN_BOOL:
```

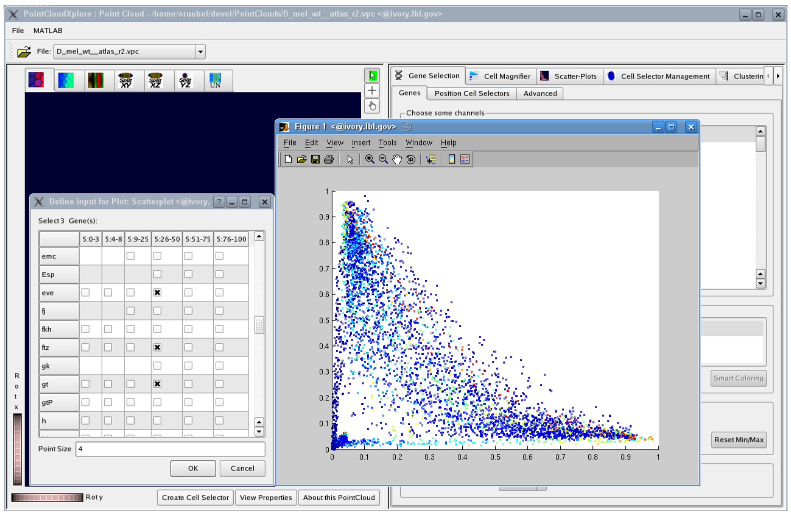


Figure A.2: **Gene expression scatter plot in MATLAB:** This example function creates a 2D scatter plot of two genes colored according to the values of a third gene. The size of the data points is defined by a user-defined input parameter.

Scatterplot.m

```
function Scatterplot(input_genes,S)
x = input_genes(1,:);
y = input_genes(2,:);
c = input_genes(3,:);
size = cast(S,'double');
scatter(x,y,S,c);
```

Scatterplot.pcxm

```
TYPE:Plot
NAME:Scatterplot
DIR:
CALL:Scatterplot(input_genes,S)
IN_GENES:3
IN_BRUSHES:
IN_DOUBLE:
IN_INT:S:Point Size
IN_BOOL:
```

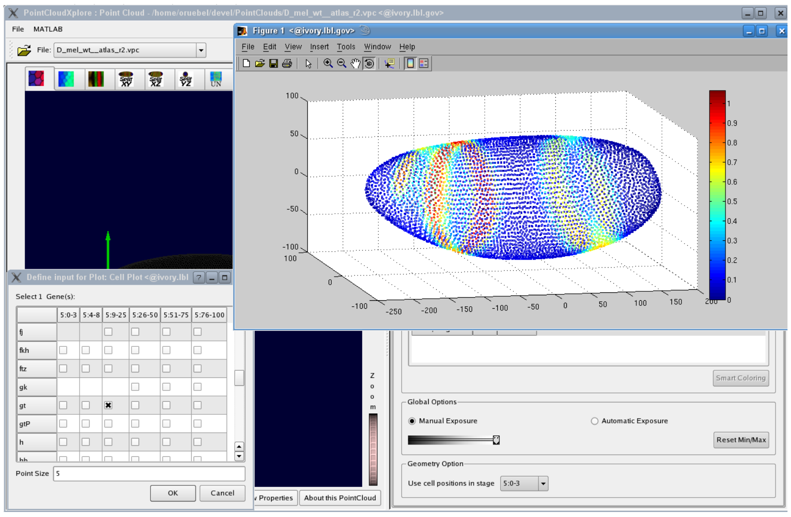


Figure A.3: **Cell position plot in MATLAB:** This example function creates a 3D scatter plot of the *x*, *y*, *z* cell positions. Color is defined with respect to a selected gene channel. The size of the data points is defined by a separate user-defined input parameter.

CellPlot.m
function CellPlot(input_cellpos , input_genes , S)
x = input_cellpos(1,:);
y = input_cellpos(2,:);
z = input_cellpos(3,:);
c = input_genes(1,:);
size = cast(S , 'double');
scatter3(x,y,z,size,c);

CellPlot.pcxm
TYPE:Plot
NAME:Cell Plot
DIR:
CALL:CellPlot(input_cellpos , input_genes , S)
IN_GENES:1
IN_BRUSHES:
IN_DOUBLE:
IN_INT:S:Point Size
IN_BOOL:

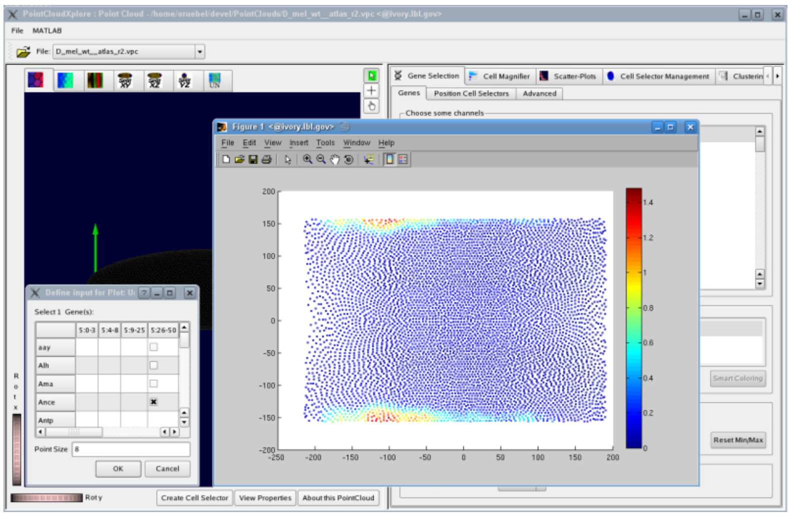


Figure A.4: **Unrolled view in MATLAB:** This example function creates a 2D unrolled point plot colored according to the expression values of a selected gene. The size of the data points is defined by a user-defined input parameter.

Unrolled.m
function Unrolled(input_genes , input_unrolledpos , S)
x = input_unrolledpos(1,:);
y = input_unrolledpos(2,:);
c = input_genes(1,:);
size = cast(S , 'double');
scatter(x,y,size,c);

Unrolled.pcxm
TYPE:Plot
NAME:Unrolled
DIR:
CALL:Unrolled(input_genes , input_unrolledpos , S)
IN_GENES:1
IN_BRUSHES:
IN_DOUBLE:
IN_INT:S:Point Size
IN_BOOL:

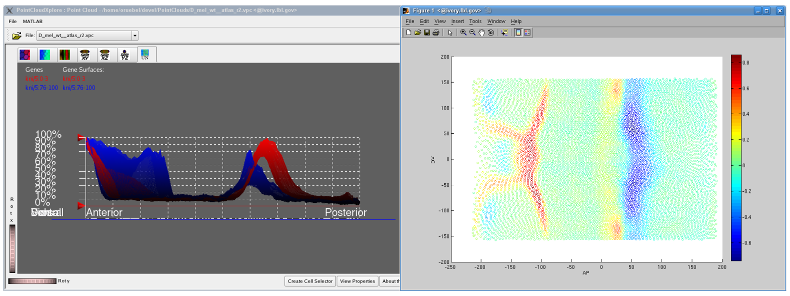


Figure A.5: **Expression difference MATLAB filter:** This simple example filter computes the difference between two input patterns and creates a new pseudo-gene-channel with the according information. In the image the two input patterns are shown via expression surfaces (left). The difference pattern is shown using the Unrolled View MATLAB script described above which is called from PCX using the newly created pseudo-gene channel as input.

Diff.m

```
function
output_genes=Diff(input_genes)
output_genes=input_genes(2,:)-
input_genes(1,:);
```

Diff.pcxm

```
TYPE:Arithmetic
NAME:Diff
DIR:
CALL:output_genes=Diff(input_genes);
IN_GENES:2
IN_BRUSHES:
IN_DOUBLE:
IN_INT:
IN_BOOL:
```

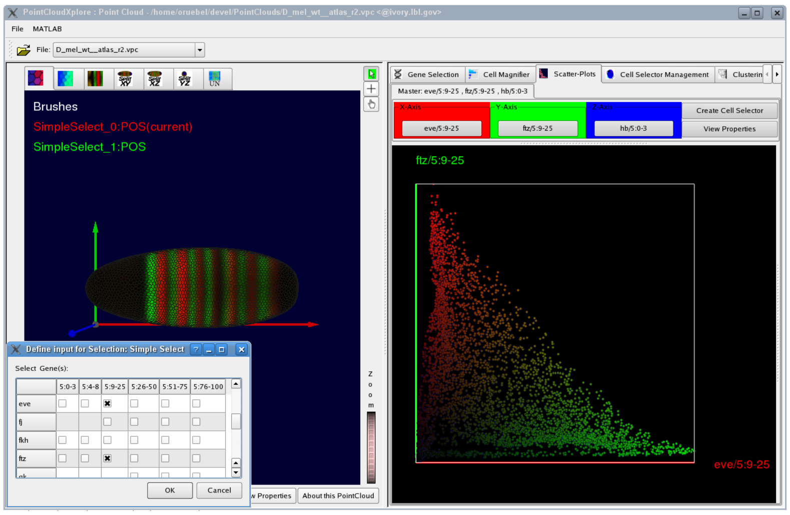


Figure A.6: **Cell selection using MATLAB:** This simple example selection function defines a set of smooth cell selection by using the input expression values to define a cell selection. Gene expression values are send as input from PCX to MATLAB. These values are then simply copied to the *output_brushes* array so that PCX then creates a set of cell selectors with the according filter values. The figure shows two example cell selectors created based on the expression of *eve* and *ftz* using this simple MATLAB function. The selections are visualized in the 3D cell view and a 2D scatter plot of *eve* and *ftz*. The window in the bottom left shows the GUI for the SimpleSelect function.

SimpleSelect.m

```
function output_brushes =
SimpleSelect(input_genes)
output_brushes = input_genes;
```

SimpleSelect.pcxm

```
TYPE:Selection
NAME:Simple Select
DIR:
CALL: output_brushes =
SimpleSelect(input_genes);
IN_GENES:
IN_BRUSHES:
IN_DOUBLE:
IN_INT:
IN_BOOL:
```

Bibliography

[1] K. W. Brodlie, L. Carpenter, R. A. Earnshaw, J. R. Gallop, R. J. Hubbold, A. M. Mumford, C. D. Osland, and P. Quarendon, Eds., *Scientific visualization: techniques and applications*. New York, NY, USA: Springer-Verlag New York, Inc., 1992.

[2] B. H. McCormick, T. A. DeFanti, and M. D. Brown, Eds., *Visualization in Scientific Computing*. Computer Graphics, November 1987, vol. 21, no. 6.

[3] HyperVis - Teaching Scientific Visualization Using Hypermedia. [Online]. Available: http://www.siggraph.org/education/materials/HyperVis/visgoals/visgoal2.htm

[4] E. Klein and O. G. Staadt, "Sonification of three-dimensional vector fields," in *Proceedings of the SCS High Performance Computing Symposium*. Society for Modeling and Simulation International, April 2004.

[5] J. C. Roberts and S. Paneels, "Where are we with haptic visualization?" in *WorldHaptics 2007, Second Joint EuroHaptics Conference, 2007 and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, Tsukuba, Japan, March 2007, pp. 316–323.

[6] C. D. Hansen and C. R. Johnson, *The Visualization Handbook*. Elsevier Academic Press, 2005.

[7] C. A. Brewer, "Color brewer," 200x. [Online]. Available: http://www.ColorBrewer.org

[8] R. Kosara, G. N. Sahling, and H. Hauser, "Linking scientific and information visualization with interactive 3d scatterplots," in *Short Communication Papers Proceedings of the 12th International Conferencein Central Europe on Computer Graphics, Visualization, and Computer Vision (WSCG)*, 2004, pp. 133–140.

[9] H. Piringer, R. Kosara, and H. Hauser, “Interactive focus+context visualization with linked 2d/3d scatterplots,” in *Proceedings of the Second International Conference on Coordinated & Multiple Views in Exploratory Visualization (CMV’04)*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 49–60.

[10] S. Bachthaler and D. Weiskopf, “Continuous scatterplots,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1428–1435, 2008.

[11] M. d’Ocagne, *Coordonnées Parallèles et Axiales: Méthode de transformation géométrique et procédé nouveau de calcul graphique déduits de la considération des coordonnées parallèlles*. Paris, France: Gauthier-Villars, 1885.

[12] A. Inselberg, “Parallel coordinates for multidimensional displays,” in *Spatial Information Technologies for Remote Sensing Today and Tomorrow, The Ninth William T. Pecora Memorial Remote Sensing Symposium*. IEEE Computer Society Press, 1984, pp. 312–324.

[13] E. J. Wegman, “Hyperdimensional data analysis using parallel coordinates,” *Journal of the American Statistical Association*, vol. 85, no. 411, pp. 664–675, Sept. 1990.

[14] A. Inselberg, H. Hauser, M. Ward, and L. Yang, “Modern parallel coordinates: from relational information to clear patterns,” in *IEEE Visualization*, October 2006, tutorial.

[15] A. Inselberg, *Parallel Coordinates Visual Multidimensional Geometry and Its Applications*. Springer-Verlag, 2008.

[16] E. Wegman and Q. Luo, “High dimensional clustering using parallel coordinates and the grand tour,” *Computing Science and Statistics*, vol. 28, pp. 361–368, 1996.

[17] M. Graham and J. Kennedy, “Using curves to enhance parallel coordinate visualisations,” in *IV ’03: Proceedings of the Seventh International Conference on Information Visualization*. Washington, DC, USA: IEEE Computer Society, 2003, p. 10.

[18] R. Moustafa and E. Wegman, “On some generalization to parallel coordinate plot,” in *Seeing a million, A Data Visualization Workshop, Rain am Lech (nr.)*, 2002.

[19] W. Peng, M. O. Ward, and E. A. Rundensteiner, “Clutter reduction in multi-dimensional data visualization using dimension reordering,” in *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS04*. IEEE Computer Society, 2004, pp. 89–96.

[20] J. Yang, M. O. Ward, E. A. Rundensteiner, and S. Huang, “Visual hierarchical dimension reduction for exploration of high dimensional datasets,” in *VISSYM ’03: Proceedings of the symposium on Data visualisation 2003*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2003, pp. 19–28.

[21] H. Hauser, F. Ledermann, and H. Doleisch, “Angular brushing of extended parallel coordinates,” in *in: Proceedings of IEEE Symposium on Information Visualization*. IEEE Computer Society Press, 2002, pp. 127–130.

[22] E. Fanea, S. Carpendale, and T. Isenberg, “An interactive 3d integration of parallel coordinates and star glyphs,” in *INFOVIS ’05: Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization*. Washington, DC, USA: IEEE Computer Society, 2005, p. 20.

[23] Y.-H. Fua, M. O. Ward, and E. A. Rundensteiner, “Hierarchical parallel coordinates for exploration of large datasets,” in *IEEE Visualization 1999*. IEEE Computer Society Press, 1999, pp. 43–50.

[24] M. Novotný, “Visually effective information visualization of large data,” in *Proceedings of Central European Seminar on Computer Graphics (CESCG)*, 2004.

[25] J. Johansson, P. Ljung, M. Jern, and M. Cooper, “Revealing structure within clustered parallel coordinates displays,” in *INFOVIS ’05: Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization*. Washington, DC, USA: IEEE Computer Society, 2005, p. 17.

[26] M. Novotný and H. Hauser, “Outlier-preserving focus+context visualization in parallel coordinates,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 893–900, 2006.

[27] L. Fahrmeir, R. Knstler, I. Pigeot, and G. Tutz, *Statistik*. Springer, Berlin, 2004, vol. 5.

[28] K. Potter, *Methods for Presenting Statistical Information: The Box Plot*, ser. GI Lecture Notes in Informatics. Bonn, Germany: Gesellschaft für Informatik (GI), 2006, vol. S-4, pp. 97–106.

[29] M. Q. W. Baldonado, A. Woodruff, and A. Kuchinsky, “Guidelines for using multiple views in information visualization,” in *AVI ’00: Proceedings of the working conference on Advanced visual interfaces*. New York, NY, USA: ACM Press, 2000, pp. 110–119.

[30] C. Henze, “Feature detection in linked derived spaces,” in *Proceedings IEEE Visualization ’98*, D. Ebert, H. Rushmeier, and H. Hagen, Eds. Los Alamitos, CA, USA: IEEE Computer Society Press, 1998, pp. 87–94.

[31] D. L. Gresh, B. E. Rogowitz, R. L. Winslow, D. F. Scollan, and C. K. Yung, “WEAVE: A system for visually linking 3-d and statistical visualizations, applied to cardiac simulation and measurement data,” in *Proceedings IEEE Visualization 2000*, T. Ertl, B. Hamann, and A. Varshney, Eds. Los Alamitos, CA, USA: IEEE Computer Society Press, 2000, pp. 489–492.

[32] H. Doleisch, M. Gasser, and H. Hauser, “Interactive feature specification for focus+context visualization of complex simulation data,” in *Data Visualization 2003 (Proceedings of the Eurographics/IEEE TCVG Symposimon Visualization)*, G.-P. Bonneau, S. Hahmann, and C. D. Hansen, Eds., 2003.

[33] SimVis. [Online]. Available: http://www.simvis.at/

[34] VisIt. [Online]. Available: https://wci.llnl.gov/codes/visit/

[35] H. Childs, E. S. Brugger, K. S. Bonnell, J. S. Meredith, M. Miller, B. J. Whitlock, and N. Max, “A contract based system for large data visualization,” in *Proceedings of IEEE Visualization 2005*, October 2005, pp. 190–198.

[36] ParaView. [Online]. Available: http://www.paraview.org/

[37] EnSight Gold. [Online]. Available: http://www.ensight.com/ensight-gold.html

[38] The VisIt user Wiki. [Online]. Available: http://visitusers.org/

[39] VisIt User’s Manual, Version 1.5, October 2005, Lawrence Livermore National Laboratory (LLNL). [Online]. Available: https://wci.llnl.gov/codes/visit/manuals.html

[40] K. Stockinger, J. Shalf, K. Wu, and E. W. Bethel, “Query-Driven Visualization of Large Data Sets,” in *Proceedings of IEEE Visualization 2005*. IEEE Computer Society Press, October 2005, pp. 167–174, LBNL-57511.

[41] R. Bellman, *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[42] E. W. Bethel, S. Campbell, E. Dart, K. Stockinger, and K. Wu, “Accelerating Network Traffic Analysis Using Query-Driven Visualization,” in *Proceedings of 2006 IEEE Symposium on Visual Analytics Science and Technology*. IEEE Computer Society Press, October 2006, pp. 115–122, LBNL-59891.

[43] K. Stockinger, E. W. Bethel, S. Campbell, E. Dart, and K. Wu, “Detecting distributed scans using high-performance query-driven visualization,” in *SC ’06: Proceedings of the 2006 ACM/IEEE Conference on High Performance Computing, Networking, Storage and Analysis*. New York, NY, USA: IEEE Computer Society Press, October 2006, LBNL-60053.

[44] D. Comer, “The ubiquitous B-tree,” *Computing Surveys*, vol. 11, no. 2, pp. 121–137, 1979.

[45] S. Chaudhuri and U. Dayal, “An overview of data warehousing and OLAP technology,” *ACM SIGMOD Record*, vol. 26, no. 1, pp. 65–74, March 1997.

[46] P. O’Neil, “Model 204 architecture and performance,” in *2nd International Workshop in High Performance Transaction Systems, Asilomar, CA*, ser. Lecture Notes in Computer Science, vol. 359. Springer-Verlag, Sept. 1987, pp. 40–59.

[47] K. Stockinger, K. Wu, and A. Shoshani, “Strategies for processing ad hoc queries on large data warehouses,” in *DOLAP’02*, McLean, Virginia, USA, 2002, pp. 72–79.

[48] K. Wu, K. Stockinger, and A. Shosani, “Breaking the curse of cardinality on bitmap indexes,” in *Scientific and Statistical Database Management Conference (SSDBM) 2008*, 2008, pp. 348–365.

[49] C.-Y. Chan and Y. E. Ioannidis, “Bitmap index design and evaluation,” in *SIGMOD Rec.*, vol. 27, no. 2. New York, NY, USA: ACM, 1998, pp. 355–366.

[50] K. Wu, E. Otoo, and A. Shoshani, “On the performance of bitmap indices for high cardinality attributes,” in *Very Large Data Base Conference (VLDB)*, 2004, pp. 24–35.

[51] FastBit. [Online]. Available: https://codeforge.lbl.gov/projects/fastbit/

[52] K. Wu, E. Otoo, and A. Shoshani, “Compressing bitmap indexes for faster search operations,” in *Scientific and Statistical Database Management Conference (SSDBM)’02*, Edinburgh, Scotland, 2002, pp. 99–108.

[53] ——, “Optimizing bitmap indices with efficient compression,” *ACM Transactions on Database Systems*, vol. 31, pp. 1–38, 2006.

[54] K. Wu, W. Koegler, J. Chen, and A. Shoshani, “Using bitmap index for interactive exploration of large datasets,” in *Scientific and Statistical Database Management Conference (SSDBM)’2003*. Washington, DC, USA: IEEE Computer Society, 2003, pp. 65–74.

[55] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Prentice Hall, 2007, vol. 3.

[56] S. W. Zucker, “Region growing: Childhood and adolescence,” *Computer Graphics and Image Processing*, vol. 5, no. 3, pp. 382 – 399, 1976.

[57] R. Adams and L. Bischof, “Seeded region growing,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 6, pp. 641–647, 1994.

[58] A. K. Jain, M. N. Murty, and P. J. Flynn, “Data clustering: A review,” in *ACM Computing Surveys*, vol. 31, no. 3, September 1999.

[59] M. J. L. de Hoon, S. Imoto, J. Nolan, and S. Miyano, “Open source clustering software,” *Bioinformatics*, vol. 20 (9), pp. 1453–1454, 2004.

[60] Y. Cheng, “Mean shift, mode seeking, and clustering,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 790–799, 1995.

[61] D. Comaniciu and P. Meer, “Mean shift: A robust approach toward feature space analysis,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 603–619, 2002.

[62] T. Kohonen, *Self-organizing maps, Edition 3*. Springer, 2001.

[63] P. Törönen, M. Kolehmainen, W. Garry, and C. Ero, “Analysis of gene expression data using self-organizing maps,” *FEBS Letters*, vol. 452,2, pp. 142–146, 1999.

[64] P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrosky, E. S. Lander, and T. R. Golub, “Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation,” in *Genetics*, vol. 96. Natl. Acad. Sci. USA, March 1999, p. 29072912.

[65] J. Handl, J. Knowles, and D. B. Kell, “Computational cluster validation in post-genomic data analysis,” *Bioinformatics*, vol. 21, no. 15, pp. 3201–3212, 2005.

[66] C. J. van Rijsbergen, *Information Retrieval , Second Edition.* Butterworths, 1979.

[67] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical Association*, vol. 66, no. 336, pp. 846–850, 1971.

[68] J. C. Dunn, "Well separated clusters and optimal fuzzy-partitions," *Journal of Cybernetics*, vol. 4, pp. 95–104, 1974.

[69] P. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of Compuational and Applied Mathematics*, vol. 20, no. 1, pp. 53–65, November 1987.

[70] K. Y. Yeung, D. R. Haynor, and W. L. Ruzzo, "Validating clustering for gene expression data," *Bioinformatics*, vol. 17, no. 4, pp. 309–318, 2001.

[71] S. Datta and S. Datta, "Comparison and validation of statistical clustering techniques for microarray gene expression data," *Bioinformatics*, vol. 19, no. 4, pp. 459–466, 2003.

[72] A. Ben-Dor, N. Friedmann, and Z. Yakhini, "Overabundance analysis and class discovery in gene expression data," Agilent Laboratories, Palo Alto, Tech. Rep., 2002.

[73] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the number of clusters in a dataset via the gap statistic," Dept. of Statistics, Stanford University, Tech. Rep. 208, 2000.

[74] G. W. Miligan and M. C. Cooper, "An examination of procedures for determining the number of clusters in a data set," *Psychometrika*, vol. 50, no. 2, pp. 159–179, June 1985.

[75] D. Jiang, C. Tang, and A. Zhang, "Cluster analysis for gene expression data: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 11, pp. 1370–1386, 2004.

[76] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein, "Cluster analysis and display of genome-wide expression patterns," in *Proc Natl Acad Sci U S*, 1995, pp. 14 863–14 868.

[77] A. A. Alizadeh, M. B. Eisen, R. E. Davis, C. Ma, I. S. Lossos, A. Rosenwald, J. C. Boldrick, H. Sabet, T. Tran, X. Yu, J. I. Powell, L. Yang, G. E. Marti, T. Moore, J. Hudson, L. Lu, D. B. Lewis, R. Tibshirani, G. Sherlock, W. C. Chan, T. C. Greiner, D. D. Weisenburger, J. O. Armitage, R. Warnke, R. Levy, W. Wilson, M. R. Grever, J. C. Byrd, D. Botstein, P. O. Brown, and L. M. Staudt, "Distinct types of diffuse large b-cell lymphoma identified by gene expression," *Nature*, vol. 403, pp. 503–511, February 2000.

[78] J. Ihmels, G. Friedlander, S. Bergmann, O. Sarig, Y. Ziv, and N. Barkai, "Revealing modular organization in the yeast transcriptional network," *Nature Genetics*, vol. 31; part 4, pp. pages 370–378, 2002.

[79] S. C. Madeira and A. L. Oliveira, "Biclustering algorithms for biological data analysis: a survey," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 1, no. 1, pp. 24–45, 2004.

[80] M. B. Eisen, cluster 2.20 and treeview 1.60, 2002. [Online]. Available: http://rana.lbl.gov/EisenSoftware.htm

[81] Spotfire., decision Site. Somerville, MA. [Online]. Available: http://www.spotfire.com

[82] M. Reich, K. Ohm, P. Tamayo, M. Angelo, and J. P. Mesirov, "Genecluster 2.0: An advanced toolset for bioarray analysis," in *Bioinformatics*, 2004.

[83] A. Saeed, V. Sharov, J. White, J. Li, W. Liang, N. Bhagabati, J. Braisted, M. Klapa, T. Currier, M. Thiagarajan, A. Sturn, M. Snuffin, A. Rezantsev, D. Popov, A. Ryltsov, E. Kostukovich, I. Borisovsky, Y. Liu, A. Vinsavich, V. Trush, and J. Quackenbush, "Tm4: a free, open-source system for microarray data management and analysis." *Biotechniques*, vol. 34, no. 2, pp. 374–378, 2003.

[84] Rosetta Biosoftware, kirkland, WA,. [Online]. Available: http://www.rosettabio.com

[85] J. Seo and B. Shneiderman, "A knowledge integration framework for information visualization," in *From Integrated Publication and Information Systems to Information and Knowledge Environments*, ser. LNCS, vol. 3379/2005. Springer Berlin / Heidelberg, 2005, pp. 207–220.

[86] T. G. Dietterich, *Ensemble Learning*. Cambridge, MA, USA: MIT Press, 2002.

[87] A. Fred and A. Jain., "Combining multiple clustering using evidence accumulation," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 6, 2005, pp. 835–850.

[88] A. Strehl and J. Ghosh, "Cluster ensembles – a knowledge reuse framework for combining multiple partitions," *Journal on Machine Learning Research (JMLR)*, vol. 3, pp. 583–617, December 2002.

[89] J. Kittler, M. Hatef, R. P. Duin, and J. Matas, “On combining classifiers,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, pp. 226–239, March 1998.

[90] T. G. Dietterich, “Ensemble methods in machine learning,” in *Lecture Notes in Computer Science*, vol. 1857/2000. Springer-Verlag, 2000, pp. 1–15.

[91] BDTNP. [Online]. Available: http://bdtnp.lbl.gov/Fly-Net/bioimaging.jsp

[92] P. A. Lawrence, *The Making of a Fly: The Genetics of Animal Design*. Blackwell Publishing, Inc., 1992.

[93] K. Weigmann, R. Klapper, T. Strasser, C. Rickert, G. Technau, H. Jäckle, W. Janning, and C. Klämbt, “Flymove a new way to look at development of drosophila.” [Online]. Available: http://flymove.uni-muenster.de/

[94] C. L. Luengo Hendriks, S. V. E. Keränen, C. C. Fowlkes, L. Simirenko, G. H. Weber, A. H. DePace, C. Henriquez, D. W. Kaszuba, B. Hamann, M. B. Eisen, J. Malik, D. Sudar, M. D. Biggin, and D. W. Knowles, “Three-dimensional morphology and gene expression in the Drosophila blastoderm at cellular resolution I: Data acquisition pipeline,” *Genome Biology*, vol. 7, no. 12, p. R123, 2006. [Online]. Available: http://genomebiology.com/2006/7/12/R123

[95] Specification of the Single PointCloud (.pce) file format. [Online]. Available: http://bdtnp.lbl.gov/Fly-Net/bioimaging.jsp?w=pceFormat

[96] Specification of the Virtual PointCloud (.vpc) file format. [Online]. Available: http://bdtnp.lbl.gov/Fly-Net/bioimaging.jsp?w=vpcFormat

[97] G. H. Weber, C. L. H. Luengo, S. V. E. Keränen, S. E. Dillard, D. Ju, D. Sudar, and B. Hamann, “Visualization for validation and improvement of three-dimensional segmentation algorithms,” in *Data Visualization 2005 (Proceedings of the EUROGRAPHICS - IEEE VGTC Symposium on Visualization 2005*. Eurographics Association, June 2005.

[98] C. C. Fowlkes, C. L. Luengo Hendriks, S. V. E. Keränen, M. D. Biggin, D. W. Knowles, D. Sudar, and J. Malik, “Registering *Drosophila* embryos at cellular resolution to build a quantitative3d map of gene expression patterns and morphology,” in *CSB 2005 Workshop on BioImage Data Minning and Informatics*, August 2005.

[99] S. V. E. Keränen, C. C. Fowlkes, C. L. Luengo Hendriks, D. Sudar, D. W. Knowles, J. Malik, and M. D. Biggin, “Three-dimensional morphology and gene expression in the Drosophila blastoderm at cellular resolution I: Dynamics,” *Genome Biology*, vol. 7, no. 12, p. R124, 2006. [Online]. Available: http://genomebiology.com/2006/7/12/R124

[100] Office of High Energy Physics, “US Particle Physics: Science Opportunities; A Strategic Plan for the Next ten Years,” DOE, Tech. Rep., May 2008. [Online]. Available: http://www.er.doe.gov/hep/HEPAP/reports/P5_Report%2006022008.pdf

[101] CERN Large Hadron Collider. [Online]. Available: http://public.web.cern.ch/public/en/LHC/LHC-en.html

[102] T. Tajima and J. M. Dawson, “Laser electron accelerator,” *Phys. Rev. Lett.*, vol. 43, no. 4, pp. 267–270, July 1979.

[103] E. Esarey, P. Sprangle, J. Krall, and A. Ting, “Overview of plasma-based accelerator concepts,” *IEEE Transactions on Plasma Science*, vol. 24, no. 2, pp. 252–288, April 1996.

[104] W. P. Leemans, B. Nagler, A. J. Gonsalves, C. Toth, K. Nakamura, C. G. R. Geddes, E. Esarey, C. B. Schroeder, and S. M. Hooker, “Gev electron beams from a centimetre-scale accelerator,” *Nature Physics*, vol. 2, pp. 696 – 699, 2006.

[105] K. Nakamura, B. Nagler, C. Toth, C. G. R. Geddes, C. B. Schroeder, E. Esarey, W. P. Leemans, A. J. Gonsalves, and S. M. Hooker, “Gev electron beams from a centimeter-scale channel guided laser wakefield accelerator,” *Physics of Plasma*, vol. 14, 056708, 2007.

[106] C. G. R. Geddes, “Plasma channel guided laser wakefield accelerator,” Ph.D. dissertation, University of California, Berkeley, 2005.

[107] V. Malka, S. Fritzler, E. Lefebvre, M.-M. Aleonard, F. Burgy, J.-P. Chambaret, J.-F. Chemin, K. Krushelnick, G. Malka, S. P. D. Mangles, Z. Najmudin, M. Pittman, J.-P. Rousseau, J.-N. Scheurer, B. Walton, and A. E. Dangor, “Electron acceleration by a wake field forced by an intense ultrashort laser pulse,” *Science 298 (5598), 1596*, November 2002.

[108] C. Geddes, C. Toth, J. van Tilborg, E. Esarey, C. Schroeder, D. Bruhwiler, C. Nieter, J. Cary, and W. Leemans, “High-Quality Electron Beams from a Laser Wakefield Accelerator Using Plasma-Channel Guiding,” *Nature*, vol. 431, pp. 538–541, 2004, LBNL-55732.

[109] J. Faure, Y. Glinec, A. Pukhov, S. Kiselev, S. Gordienko, E. Lefebvre, J.-P. Rousseau, F. Burgy, and V. Malka, "A laserplasma accelerator producing monoenergetic electron beams," *Nature*, vol. 431, pp. 541–544, 2004.

[110] S. P. D. Mangles, C. D. Murphy, Z. Najmudin, A. G. R. Thomas, J. L. Collier, A. E. Dangor, E. J. Divall, P. S. Foster, J. G. Gallacher, C. J. Hooker, D. A. Jaroszynski, A. J. Langley, W. B. Mori, P. A. Norreys, F. S. Tsung, R. Viskup, B. R. Walton, and K. Krushelnick, "Monoenergetic beams of relativistic electrons from intense laserplasma interactions," *Nature*, vol. 431, pp. 535–538, 2004.

[111] Lasers, Optical Accelerator Systems Integrated Studies (LOASIS) Program. [Online]. Available: http://loasis.lbl.gov/

[112] C. G. R. Geddes, K. Nakamura, G. R. Plateau, C. Toth, E. C. Michel, E. Asarey, C. B. Schroeder, J. R. Cary, and W. P. W. P. Leemans, "Plasma-density-gradient injection of low absolute-momentum-spread electron bunches," *Physical Review Letters (PRL)*, vol. 100, 215004, 2008.

[113] F. S. Tsung, R. Narang, W. B. Mori, C. Joshi, R. A. Fonseca, and L. O. Silva, "Near-gev-energy laser-wakefield acceleration of self-injected electrons in a centimeter-scale plasma channel," *Physical Review Letters (PRL)*, vol. 93, 185002, October 2004.

[114] C. G. R. Geddes, D. Bruhwiler, J. R. Cary, E. Cormier-Michel, E. Esarey, C. B. Schroeder, W. A. Isaacs, N. Stinus, P. Messmer, A. Hakim, K. Nakamura, A. J. Gonsalves, D. Panasenko, G. R. Plateau, C. Toth, B. Nagler, J. van Tilborg, T. Cowan, S. M. Hooker, , and W. P. Leemans, "Laser wakefield simulations towards development of compact particle accelerators," in *Journal of Physics: Conference Series 78 , SciDAC 2007 IOP Publishing*, 2007.

[115] C. G. R. Geddes, D. L. Bruhwiler, J. R. Cary, W. B. Mori, J.-L. Vay, S. F. Martins, T. Katsouleas, E. Cormier-Michel1, W. M. Fawley, C. Huang, X. Wang, B. Cowan, V. K. Decyk, E. Esarey, R. A. Fonseca, W. Lu, P. Messmer, P. Mullowney, K. Nakamura, K. Paul, G. R. Plateau, C. B. Schroeder, L. O. Silva, C. Toth, F. S. Tsung, M. Tzoufras, T. Antonsen, J. Vieira, and W. P. Leemans, "Computational studies and optimization of wakefield accelerators," in *Journal of Physics: Conference Series 125 012002, SciDAC 2008 IOP Publishing*, 2008.

[116] E. Michel, B. A. Shadwick, C. B. Schroeder, C. G. R.Geddes, E. Esarey, W. P. Leemans, H. Ruhl, and T. Cowan, “Accurate modeling of laser-plasma accelerators with particle-in-cell codes,” in *Advanced Accelerator Concepts: 12th Advanced Accelerator Concepts Workshop*, ser. American Institute of Physics Conference Series, M. Conde and C. Eyberger, Eds., vol. 877, Nov. 2006, pp. 213–219.

[117] C. Nieter and J. R. Cary, “VORPAL: A Versatile Plasma Simulation Code,” *J. Comput. Phys.*, vol. 196, no. 2, pp. 448–473, 2004.

[118] R. A. Fonseca, L. O. Silva, F. S. Tsung, V. K. Decyk, W. Lu, C. Ren, W. B. Mori, S. Deng, S. Lee, T. Katsouleas, and J. C. Adam, “Osiris: A three-dimensional, fully relativistic particle in cell code for modeling plasma based accelerators,” *Lecture Notes in Computer Science, Computational Science ICCS 2002*, vol. 2331/2002, pp. 342–351, 2002.

[119] J. Wang, P. Liewer, and V. Decyk, “3d electromagnetic plasma particle simulations on a mimd parallel computer,” *Computer Physics Communications*, vol. 87, pp. 35–53, 1995.

[120] HDF5. [Online]. Available: http://www.hdfgroup.org/HDF5/index.html

[121] Vizschema. [Online]. Available: https://ice.txcorp.com/trac/vizschema

[122] L. Gosink, J. Shalf, K. Stockinger, K. Wu, and E. W. Bethel, “HDF5-FastQuery: Accelerating Complex Queries on HDF Datasets using Fast Bitmap Indices,” in *Proceedings of the 18th International Conference on Scientific and Statistical Database Management*. IEEE Computer Society Press, July 2006, LBNL-59602.

[123] A. Adelmann, R. Ryne, J. Shalf, and C. Siegerist, “H5part: A portable high performance parallel data interface for particle simulations,” in *Particle Accelerator Conference (PAC05)*, May 16-20 2005.

[124] A. Adelmann, A. Gsell, B. Oswald, T. Schietinger, E. W. Bethel, J. Shalf, C. Siegerist, and K. Stockinger, “Progress on h5part: A portable high performance parallel data interface for electromagnetic simulations,” in *Particle Accelerator Conference PAC07*, June 2007.

[125] Root. [Online]. Available: http://root.cern.ch/drupal/

[126] PAW. [Online]. Available: http://paw.web.cern.ch/paw/

[127] AIDA. [Online]. Available: http://aida.freehep.org/

[128] R. [Online]. Available: http://www.r-project.org/

[129] IDL. [Online]. Available: http://www.ittvis.com/ProductServices/IDL.aspx

[130] OpenDX. [Online]. Available: http://www.opendx.org/

[131] VorpalView. [Online]. Available: http://www.txcorp.com/products/VORPAL/vorpalview.php

[132] R. A. Fonseca, S. F. Martins, L. O. Silva, J. W. Tonge, F. S. Tsung, and W. B. Mori, "One-to-one direct modeling of experiments and astrophysical scenarios: pushing the envelope on kinetic plasma simulations," *Plasma Physics and Controlled Fusion*, vol. 50, 124034, November 2008.

[133] S. F. Martins, R. A. Fonseca, L. O. Silva, and W. B. Mori, "On dynamics and acceleration in relativistic shocks," *Astrophysical Journal Letters (ApJ)*, vol. 695, L189-L193, April 2009.

[134] A. Bagherjeiran and C. Kamath, "Graph-based methods for orbit classification," in *SDM*, 2006.

[135] N. S. Love and C. Kamath, "Image analysis for the identification of coherent structures in plasma," in *Applications of Digital Image Processing; Proceedings of the SPIE*, Tescher and A. G., Eds., vol. 6696, 2007.

[136] J. Hlina, V. Nwnicka, and J. Sonsky, "Identification of dynamic patterns and their velocities in thermal plasma jets," *Czechoslovak Journal of Physics*, vol. 54, no. 2, pp. 199–210, Feburary 2004, iSSN0011-4626 (Print) 1572-9486 (Online).

[137] **O. Rübel**, M.-Y. Huang, and M. Biggin, "Pointcloudxplore 2 online user manual." [Online]. Available: http://bdtnp.lbl.gov/Fly-Net/pcx.jsp?w=vis

[138] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics, Principles and Practice*, 2nd ed. Addison-Wesley, 1997, ch. 19.5.2.

[139] P. L. Rosin, "Unimodal thresholding," *Pattern Recognition*, vol. 34, no. 11, pp. 2083–2096, 2001.

[140] D. Berrar, W. Dubitzky, and M. Granzow, "Singular value decomposition and principal component analysis," in *In A Practical Approach to Microarray Data Analysis*. Kluwer: Norwell, MA, 2003, pp. 91–109.

[141] O. Alter, P. O. Brown, , and D. Botstein, “Processing and modeling genome-wide expression data using singular value decomposition,” in *Microarrays: Optical Technologies and Informatics*, M. L. Bittner, Y. Chen, A. N. Dorsel, and E. R. Dougherty, Eds., vol. 4266, 2001.

[142] H. Janssens, S. Hou, J. Jaeger, A.-R. Kim, E. Myasnikova, D. Sharp, and J. Reinitz, “Quantitative and predictive model of transcriptional control of the drosophila melanogaster even skipped gene,” *Nature Genetics*, vol. 38, no. 10, pp. 1159–1165, 2006.

[143] R. Albert and H. Othmer, “The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in drosophila melanogaster.” *J Theor Biol*, vol. 223(1), pp. 1–18., July 2003.

[144] L. Sanchez and D. Thieffry, “A logical analysis of the drosophila gap-gene system,” *Journal of Theoretical Biology.*, vol. 211(2), pp. 115–141, 2001.

[145] J. Jaeger, S. Surkova, M. Blagov, H. Janssens, D. Kosman, K. N. Kozlov, Manu, E. Myasnikova, C. E. Vanario-Alonso, M. Samsonova, D. H. Sharp, and J. Reinitz, “Dynamic control of positional information in the early drosophila embryo,” *Nature*, vol. 430, pp. 368 – 371, 2004. [Online]. Available: http://dx.doi.org/10.1371%2Fjournal.pcbi.0020051

[146] S. Guha, K. Shim, and J. Woo, “Rehist: relative error histogram construction algorithms,” in *VLDB ’04: Proceedings of the Thirtieth international conference on Very large Data Bases*. VLDB Endowment, 2004, pp. 300–311.

[147] E. Kreyszig, *Advanced Engineering Mathematics*. John Wiley and Sons, Ltd, 2006.

[148] B. W. Silverman, *Density Estimation for statistics and data analysis*, ser. Monographs on Statistics and Applied Probability 26. London: Chapman and Hall, 1986.

[149] C. F. Bazlamaçci and K. S. Hindi, “Minimum-weight spanning tree algorithms a survey and empirical study,” *Comput. Oper. Res.*, vol. 28, no. 8, pp. 767–785, 2001.

[150] R. O. Duda, P. Hart, and D. G. Stork, *Pattern Classification*. John Wiley and Sons, Ltd, 2001.

[151] L. Kaufman and P. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley and Sons, Ltd, 1990.

[152] C. L. F and M. J. R. C, *Shape analysis and classification:Theory and Practice*. CRC Press, Boca Raton, 2001.

[153] R. Dave and S. Sen, “Robust fuzzy clustering of relational data,” *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 6, pp. 713–727, 2002.

[154] M. J. Crawley, *The R Book*. John Wiley and Sons, Ltd, 2007.

[155] Kitware, Inc., *VTK User’s Guide, 5th Edition*. Kitware, Inc., September 2006.

[156] C.-R. Lin, K.-H. Liu, and M.-S. Chen, “Dual clustering: integrating data clustering over optimization and constraint domains,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 5, pp. 628– 637, May 2005.

List of Publications

[157] **O. Rübel**, G. H. Weber, S. V. E. Keränen, C. C. Fowlkes, C. L. L. Hendriks, L. Simirenko, N. Y. Shah, M. B. Eisen, M. D. Biggin, H. Hagen, J. D. Sudar, J. Malik, D. Knowles, and B. Hamann., "PointCloudXplore: Visual analysis of 3D gene expression data using physical views and parallel coordinates," in *Data Visualization 2006 (Proceedings of EuroVis 2006)*, B. S. Santos, T. Ertl, and K. Joy, Eds. Aire-la-Ville, Switzerland: Eurographics Association, 2006, pp. 203–210. [Online]. Available: http://vis.lbl.gov/Publications/2006/LBNL-60005.pdf

[158] **O. Rübel**, Prabhat, K. Wu, H. Childs, J. Meredith, C. G. R. Geddes, E. Cormier-Michel, S. Ahern, G. H. weber, P. Messmer, H. Hagen, B. Hamann, and E. W. Bethel, "High Performance Multivariate Visual Data Exploration for Extremely Large Data," in *SuperComputing 2008 (SC08)*, Austin, Texas, USA, November 2008. [Online]. Available: http://vis.lbl.gov/Publications/2008/LBNL-716E.pdf

[159] G. H. Weber, **O. Rübel**, M.-Y. Huang, A. H. DePace, C. C. Fowlkes, S. V. Keränen, C. L. Luengo Hendriks, H. Hagen, D. W. Knowles, J. Malik, M. D. Biggin, and B. Hamann, "Visual Exploration of Three-dimensional Gene Expression using Physical Views and Linked Abstract Views," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 6, no. 2, pp. 296–309, 2009. [Online]. Available: http://vis.lbl.gov/Publications/2008/LBNL-63776.pdf

[160] **O. Rübel**, "Integrating Data Analysis and Visualization for the Exploration of Three-dimensional Gene Expression Data," Master's thesis, University of Kaiserslautern, January 2006.

[161] C. C. Fowlkes, C. L. L. Hendriks, S. V. E. Keränen, G. H. Weber, **O. Rübel**, M.-Y. Huang, S. Chatoor, A. H. DePace, L. Simirenko, C. Henriquez, A. Beaton, R. Weiszmann, S. Celniker, B. Hamann, D. W. Knowles, M. D. Biggin, M. B.

Eisen, and J. Malik., “A Quantitative Spatio-temporal Atlas of Gene Expression in the Drosophila Blastoderm,” *Cell*, vol. 133, pp. 364–374, April 2008. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S009286740800281X

[162] **O. Rübel**, “Using Linked Scatterplots and 3D Embryo Views for Visualization of 3D Gene Expression Data in Drosophila Melanogaster,” Master’s thesis, University of Kaiserslautern, June 2005, project thesis.

[163] M.-Y. Huang, **O. Rübel**, G. H. Weber, C. L. Luengo Hendriks, M. D. Biggn, H. Hagen, and B. Hamann, *Segmenting Gene Expression Patterns of Early-stage Drosophila Embryos*. Heidelberg, Germany: Springer Verlag, 2008, pp. 313–327. [Online]. Available: http://vis.lbl.gov/Publications/2008/LBNL-62450.pdf

[164] **O. Rübel**, G. Weber, S. V. E. Keränen, C. C. Fowlkes, C. L. L. Hendriks, L. Simirenko, N. Y. Shah, M. B. Eisen, M. D. Biggin, H. Hagen, D. Sudar, J. Malik, D. W. Knowles., and B. Hamann, *PointCloudXplore: A Visualization Tool for 3D Gene Expression Data*, ser. GI Lecture Notes in Informatics. Bonn, Germany: Gesellschaft für Informatik (GI), 2006, vol. S-4, ch. PART II: Visualization of Application Data, pp. 107–117. [Online]. Available: http://vis.lbl.gov/Publications/2006/LBNL-62336.pdf

[165] E. W. Bethel, C. Johnson, C. Aragon, Prabhat, **O. Rübel**, G. Weber, V. Pascucci, H. Childs, P.-T. Bremer, B. Whitlock, S. Ahern, J. Meredith, G. Ostrouchov, K. Joy, B. Hamann, C. Garth, M. Cole, C. Hansen, S. Parker, A. Sanderson, C. Silva, and X. Tricoche, “DOE’s SciDAC Visualization and Analytics Center for Enabling Technologies – Strategy for Petascale Visual Data Analysis Success,” *CTWatch Quarterly*, vol. 3, no. 4, pp. 32–40, 2007. [Online]. Available: http://vis.lbl.gov/Publications/2007/vacet-Oct2007-CTWatchQuarterly.pdf

[166] **O. Rübel**, G. H. Weber, M.-Y. Huang, E. W. Bethel, S. V. E. Keränen, C. C. Fowlkes, C. L. L. Hendriks, A. H. DePace, L. Simirenko, M. B. Eisen, M. D. Biggin, H. Hagen, J. Malik, D. W. Knowles, and B. Hamann, *PointCloudXplore 2: Visual Exploration of 3D Gene Expression*, ser. GI Lecture Notes in Informatics. Gesellschaft fuer Informatik (GI), 2008. [Online]. Available: http://vis.lbl.gov/Publications/2008/LBNL-249E.pdf

[167] D. Ushizima, **O. Rübel**, Prabhat, G. Weber, E. Bethel, C. Aragon, C. Geddes, E. Cormier-Michel, B. Hamann, P. Messmer, and H. Hagen, “Automated analysis for detecting beams in laser wakefield simulations,” in *Seventh International Conference*

on Machine Learning and Applications (ICMLA'08), San Diego, CA, December 2008. [Online]. Available: http://vis.lbl.gov/Publications/2008/LBNL-960E.pdf

[168] E. W. Bethel, **O. Rübel**, Prabhat, K. Wu, G. H. Weber, V. Pascucci, H. Childs, A. Mascarenhas, J. Meredith, and S. Ahern, "Modern Scientific Visualization is More than Just Pretty Pictures," in *Numerical Modeling of Space Plasma Flows : Astronum-2008 (Astronomical Society of the Pacific Conference Series).*, 2009. [Online]. Available: http://vis.lbl.gov/Publications/2009/LBNL-1450E.pdf

[169] **O. Rübel**, G. H. Weber, M.-Y. Huang, E. W. Bethel, M. D. Biggin, C. C. Fowlkes, C. L. Hendriks, S. V. E. Keränen, M. B. Eisen, D. W. Knowles, J. Malik, H. Hagen, and B. Hamann, "Integrating Data Clustering and Visualization for the Analysis of 3D Gene Expression Data," *IEEE Transactions on Computational Biology and Bioinformatics*, 2009, to appear. [Online]. Available: http://vis.lbl.gov/Publications/2008/LBNL-382E-TCBB.pdf

[170] **O. Rübel**, C. G. Geddes, E. Cormier-Michel, K. Wu, Prabhat, G. H. Weber, D. M. Ushizima, P. Messmer, H. Hagen, B. Hamann, and W. Bethel, "Automatic Beam Path Analysis of LaserWakefield Particle Acceleration Data," p. 015005 (38pp), Nov. 2009. [Online]. Available: http://www.iop.org/EJ/article/1749-4699/2/1/015005/csd9_1_015005.pdf

[171] E. W. Bethel, C. Johnson, S. Ahern, J. Bell, P.-T. Bremer, H. Childs, E. Cormier-Michel, M. Day, E. Deines, T. Fogal, C. Garth, C. G. R. Geddes, H. Hagen, B. Hamann, C. Hansen, J. Jacobsen, K. Joy, J. Krüger, J. Meredith, P. Messmer, G. Ostrouchov, V. Pascucci, K. Potter, Prabhat, D. Pugmire, **O. Rübel**, A. Sanderson, C. Silva, D. Ushizima, G. H. Weber, B. Whitlock, and K. Wu, "Occam's razor and petascale visual data analysis," in *Proceedings of SciDAC 2009*, 2009. [Online]. Available: http://vis.lbl.gov/Publications/2009/LBNL-2210E.pdf

[172] K. Wu, S. Ahern, E. W. Bethel, J. Chen, H. Childs, E. Cormier-Michel, C. G. R. Geddes, J. Gu, H. Hagen, B. Hamann, W. Koegler, J. Lauret, J. Meredith, P. Messmer, E. Otoo, V. Perevoztchikov, A. Poskanzer, Prabhat, **O. Rübel**, A. Shoshani, A. Sim, K. Stockinger, G. H. Weber, and W.-M. Zhang, "FastBit: Interactively Searching Massive Data," in *Proceedings of SciDAC 2009*, 2009. [Online]. Available: http://vis.lbl.gov/Publications/2009/LBNL-2164E.pdf

[173] D. M. Ushizima, C. G. R. Geddes, E. Cormier-Michel, E. W. Bethel, J. Jacobsen, Prabhat, **O. Rübel**, G. H. Weber, P. Messmer, B. Hamann, and H. Hagen, *Automated detection and analysis of particle beams in laser-plasma accelerator simulations*. Vienna, Austria: In-Tech, 2009, ISBN 978-953-7619-X-X.

[174] G. H. Weber, S. Ahern, E. W. Bethel, S. Borovikov, H. R. Childs, E. Deines, C. Garth, H. Hagen, B. Hamann, K. I. Joy, D. Martin, J. Meredith, Prabhat, D. Pugmire, **O. Rübel**, B. V. Straalen, and K. Wu, "Recent Advances in VisIt: AMR Streamlines and Query-Driven Visualization," in *Numerical Modeling of Space Plasma Flows : Astronum-2009*, vol. 407, 2010, *to appear.*

[175] **O. Rübel**, S. V. E. Keränen, M. D. Biggin, D. W. Knowles, G. H. Weber, H. Hagen, B. Hamann, and E. W. Bethel, *Linking Advanced Visualization and MATLAB for the Analysis of 3D Gene Expression Data*. Heidelberg, Germany: Springer Verlag, 2009, *submitted.*